NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

NETWORK SHORTEST PATH APPLICATION FOR OPTIMUM TRACK SHIP ROUTING

by

Anel A. Montes

June 2005

Thesis Advisor:	Gerald G. Brown
Second Reader:	W. Matthew Carlyle

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington DC 20503.			
1. AGENCY USE ONLY ***(Leave blank)***	**2. REPORT DATE** June 2005	**3. REPORT TYPE AND DATES COVERED** Master's Thesis	
4. TITLE AND SUBTITLE: Network Shortest Path Application for Optimum Track Ship Routing			**5. FUNDING NUMBERS**
6. AUTHOR(S) Anel A. Montes			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			**8. PERFORMING ORGANIZATION REPORT NUMBER**
9. SPONSORING /MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.			
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release; distribution is unlimited			**12b. DISTRIBUTION CODE**

13. ABSTRACT (maximum 200 words)

The United States Navy Meteorology and Oceanography (METOC) community routes ships for weather evasion using advanced meteorological modeling and satellite data, but lacks a tool to enable fewer ship routers to make better routing decisions faster. Limited resources and rising costs are impacting the frequency and duration of current naval operations. The Commander, Naval Meteorology and Oceanography Command has ordered the community to find efficiencies and automation possibilities in order to meet lower manning levels, reduce waste, and increase savings. Outside of the Navy, Ocean Systems Incorporated in Alameda, CA developed the Ship Tracking and Routing System (STARS) software package to calculate optimum sea routes based on weather model data. However, METOC ship routers are reluctant to adopt this complex software. To help solve this, we modeled Optimum Track Ship Routing (OTSR) for U.S. Navy warships using a network graph of the Western Pacific Ocean. A binary heap version of Dijkstra's algorithm determines the optimum route given model generated wind and seas input. We test the model against recent weather data to verify the model's performance, and to historical divert route recommendations in order to validate against routes developed by OTSR personnel.

14. SUBJECT TERMS Networks, Dijkstra, Shortest Path, Optimum Track Ship Routing			**15. NUMBER OF PAGES** 93
			16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	**18. SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	**19. SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	**20. LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

NETWORK SHOREST PATH APPLICATION FOR OPTIMUM TRACK SHIP ROUTING

Anel A. Montes
Lieutenant, United States Navy
B.S., United States Naval Academy, 1993

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 2005

Author: Anel A. Montes

Approved by: Gerald G. Brown
Thesis Advisor

W. Matthew Carlyle
Second Reader

James N. Eagle
Chairman, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

The United States Navy Meteorology and Oceanography (METOC) community routes ships for weather evasion using advanced meteorological modeling and satellite data, but lacks a tool to enable fewer ship routers to make better routing decisions faster. Limited resources and rising costs are impacting the frequency and duration of current naval operations. The Commander, Naval Meteorology and Oceanography Command has ordered the community to find efficiencies and automation possibilities in order to meet lower manning levels, reduce waste, and increase savings. Outside of the Navy, Ocean Systems Incorporated in Alameda, CA developed the Ship Tracking and Routing System (STARS) software package to calculate optimum sea routes based on weather model data. However, METOC ship routers are reluctant to adopt this complex software. To help solve this, we modeled Optimum Track Ship Routing (OTSR) for U.S. Navy warships using a network graph of the Western Pacific Ocean. A binary heap version of Dijkstra's algorithm determines the optimum route given model generated wind and seas input. We test the model against recent weather data to verify the model's performance, and to historical divert route recommendations in order to validate against routes developed by OTSR personnel.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF FIGURES

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would first and foremost like to thank Dr. Gerald Brown and Dr. Matthew Carlyle for lending their expert knowledge and selfless mentorship. The education they have provided is invaluable. I would also like to thank Dr. Jeffrey Lerner, Mr. Paul Wittmann, and Mr. William Anderson for their efforts in obtaining and formatting the numerical weather data. This thesis would not have been possible without them. Last, but not least, I would like to thank all those at NAVPACMETOCCEN Pearl Harbor who took the time out of their busy schedules to assist with this research.

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

Navy METOC Officers and Aerographer Mates (AG) formulate optimum ship routes based on climatology, numerical weather forecasts, satellite products, and the individual ship's sailing capabilities. Even though scientific weather analysis and forecasting techniques are used to predict wind velocities and significant wave heights, an actual diversion route to avoid severe weather is developed by a team of personnel using manual techniques that take time and effort. Also, it is possible that a diversion route chosen by routing personnel may not be optimal due to the lack of ability to conduct a complete analysis of alternatives. Because time and labor consuming techniques are used, there is currently no efficient way to test different routes by varying ship characteristics and exogenous weather data.

The METOC community performs OTSR effectively. There is, however, ample room to automate the process and add tools that can quickly develop routes that may be more desirable. Commanding officers and shipmasters often ask: “why are you sending me here?” or “do I really have to go that far off my present track?” The decision maker on the ship has to feel comfortable that the person making the recommendation has taken into consideration not only the ship's safety, but economic and operational concerns as well. OTSR duty officers and the supporting members of the METOC watch team take these factors into consideration, but their answers may not be convincing enough because there is not enough time or there is no tool available to allow a full analysis when deciding an optimal route. The manual methods used to formulate the diversion are adequate to get the ship “out of trouble”, but, if several optimal tracks based on shortest path, least time, least cost, or any combination of these factors can be analyzed, time and fuel savings may be possible at no additional risk to the vessel. A tool that quickly generates tracks based on various inputs will offer routing personnel the ability to analyze the parameter sensitivities and analysis of alternatives of these tracks more fully than is currently possible with manual methods.

Commander, Naval Meteorology and Oceanography Command is responsible for meeting the Navy's OTSR requirement. He has announced that the safe operation of maritime forces through the avoidance of severe weather still remains paramount. But with limited resources challenging the routing services and rising costs impacting naval operations, optimal routing will become increasingly more important. Due to this, the METOC community is under a mandate to operate as a fully efficient and effective enterprise. For the safety of maritime assets, this includes the consolidation and automation of OTSR services. Since 1993, Ocean Systems Incorporated has been producing the Vessel Optimization and Safety System and the Ship Tracking and Routing System software in order to assist commercial routing services by generating optimal ship voyage tracks based on ship and weather inputs. These software packages were introduced to the METOC routing services in an attempt to assist in the development of route recommendations and diversions. However, routing personnel chose not to use the software due to the design favoring commercial merchant vessels and difficulty of use.

This thesis will build on the concept of using shortest paths models to analyze ship routing and provide an operations research tool to aid in the planning for initial route recommendations and diversions. It will enable METOC personnel to provide decision makers multiple route recommendations quickly, based on a more complete and thorough analysis. Second, it will show that a time-tested proven algorithm for solving shortest paths used in conjunction with simple ship response functions can be easily automated and sufficient. The model for this thesis covers the Western Pacific. We choose this theater of operation due to its political and operational importance.

This thesis incorporates numerical weather predictions into a shortest path network flow model in order to provide optimal tracks around adverse weather. The network is constructed of nodes and arcs overlying the Western Pacific Ocean. Nodes represent latitude and longitude positions, while the arcs connecting them represent navigable paths between nodes. Each node is populated with distance and time labels, wind and seas forecast predictions, and whether it is geographically accessible. It assumes the effects of currents, fog, and precipitation are negligible. Arcs “costs” are

defined as the nonnegative distances between nodes. We use a modified Dijkstra shortest path algorithm customized to solve these problems.

This thesis is a proof of concept to form the basis for possible testing in the future. It will not attempt to fully develop a fleet ready OTSR application, but rather demonstrate that this is a tool to analyze parameter sensitivities (speed, track, ship limits) when planning diverts as well as conducting analysis of alternatives. In order to objectively assess the model, test cases using recent model data and historical case comparisons are run. The results of these tests show that we have successfully devised a network ship routing model that solves optimal path problems using a modified version of Dijkstra's shortest path algorithm and a basic ship response function. The model avoids adverse weather and solves the least-time path to a destination. It calculates useful time, distance, and fuel consumption metrics in order to quantify routing decisions. The model also demonstrates that manual routing techniques involving numerous calculations and chart plotting can be automated and solutions generated in milliseconds. We have identified how the results could be used by ship routing personnel to assist in analyzing alternatives and aiding ship routing decisions. The model's performance against the historical cases reaffirms the model's skill and gives insight to improvements that can be made in the future.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

December 17-18, 1944 was one of the deadliest periods for American naval forces in World War II (Department of the Navy, 2004). The fact that these losses did not come from combat action makes them even harder to fathom. Admiral Halsey's Task Force 38, which included seven fleet and six light carriers, eight battleships, 15 cruisers, and about 50 destroyers, was operating approximately 300 nautical miles east of Luzon in the Philippine Sea. They had just completed three days of combat against Japanese air defenses in support of upcoming amphibious operations on Mindoro and were now enroute to a rendezvous point for refueling and pursuit of Japanese survivors from the Leyte Gulf battle. During this transit, Task Force 38 encountered rapidly deteriorating weather. It became clear that the Task Force was being overtaken by a tropical storm. Halsey's staff was caught unaware and was confused as to the intensity, position, and track of this storm. Due to the lack of warning, storm avoidance was impossible. Many ships in the Task Force encountered the eye where peak winds with gusts were estimated between 130-140 knots, and sea heights in excess of 30-35 feet. Most ships reported rolls of 40 degrees with some as high as 72 degrees.

At approximately 1500 on December 18, the sustained wind began to settle below 60 knots with seas abating to 18 feet. Losses in the aftermath were staggering. Three destroyers: USS Hull, USS Spence, and USS Monaghan, capsized with a loss of all hands. A cruiser, five aircraft carriers, and three destroyers suffered severe damage. 146 aircraft were lost or damaged beyond repair. The human cost totaled 800 officers and men killed with another 80 seriously injured, making this the deadliest weather event in U.S. Navy history.

As a result of such devastation, the Pacific Command established weather stations in the Caroline Islands, and later in Manila, Iwo Jima, and Okinawa. Central weather offices for disseminating data were instituted in Guam and Leyte. These weather offices and detachments were charged with providing weather advisories and warnings to ships operating in the Pacific. This was the start of weather routing in the U.S. Navy.

The United States Navy Meteorology and Oceanography (METOC) community is responsible for providing timely and accurate weather routing recommendations to U.S. Department of Defense ships. With billion-dollar combatants, expensive cargos, and rising energy costs, every effort must be made to avoid or reduce the effects of adverse weather and sea states. Such conditions can cause damage, severe speed reductions, and lost time or money. The METOC community keeps ships out of adverse weather by providing their routing advisory service called Optimum Track Ship Routing (OTSR) (Naval METOC Command Instruction 3140.1L).

It wasn't until the mid-1950's that weather routing started to make significant contributions. Improvements in long-range forecasting, climatology, and numerical weather prediction added greatly to its viability and success. At present, satellite communications and computer web support make this service available to just about every vessel in near real time. Weather routing services come in the form of several recommendations and advisories. These include an initial route recommendation, surveillance, advisory, and diversion. These services are used to minimize weather impacts on shipping in order to maximize fuel savings, reduce damage to ship and cargo, and increase safety for crew. These metrics define optimality in ship routing.

Aside from these intuitive benefits, proof of the success and importance of OTSR comes from the demand for its services. In 2000, the Naval Pacific Meteorology and Oceanography Center (NAVPACMETOCCEN) in Yokosuka Japan, for the Western Pacific and Indian Ocean, provided over 750 successful ship surveillances, 150 diversions, and 7 sortie-from-port recommendations. Conversely, cost and safety benefits can be seen when such services are ignored or not provided. On May 21, 2001, the SS Cape Mohican (T-AKR 5065) was participating in an exercise at Chilpo Beach, just north of Pohang, South Korea. While in the harbor, heavy winds caused her to drag anchor approximately 150 feet and run aground on rocks, causing damage to her hull and internal tanks (Department of the Army, 2004). This damage required salvage work and emergency dry-docking along with 1500 tons of steel to make her seaworthy again (American Society of Naval Engineers, 2004). A similar case occurred in August 27, 2001, when the USS Greenville grounded on approach to the port of Guam. Heavy winds and seas were sited as causes to her grounding. Although she did not suffer any

extensive damage, her commanding officer was relieved. There are also cases of shipmasters not following the recommended diversion and suffering damage, not as severe as the first two examples, but suffering a cost nonetheless.

Navy METOC Officers and Aerographer Mates (AG) formulate diversion routes and ocean voyages based on climatology, numerical weather forecasts, satellite products, and the individual ship's sailing capabilities. Even though scientific weather analysis and forecasting techniques are used to predict wind velocities and significant wave heights, an actual diversion route to avoid severe weather is developed by a team of personnel using manual techniques that take time and effort. Also, it is possible that a diversion route chosen by routing personnel may not be optimal due to the lack of ability to conduct a complete analysis of alternatives. Because time and labor consuming techniques are used, there is currently no efficient way to test different routes by varying ship characteristics and exogenous weather data.

This is an important issue. As seen from the large number of successful weather routings in the Western Pacific, the METOC community performs OTSR effectively. There is, however, ample room to automate the process and add tools that can quickly develop routes that may be more desirable. Commanding officers and shipmasters often ask: “why are you sending me here?” or “do I really have to go that far off my present track?” The decision maker on the ship has to feel comfortable that the person making the recommendation has taken into consideration not only the ship's safety, but economic and operational concerns as well. OTSR duty officers and the supporting members of the METOC watch team take these factors into consideration, but their answers may not be convincing enough because there is not enough time or there is no tool available to allow a full analysis when deciding an optimal route. The manual methods used to formulate the diversion are adequate to get the ship “out of trouble”, but, if several optimal tracks based on shortest path, least time, least cost, or any combination of these factors can be analyzed, time and fuel savings may be possible at no additional risk to the vessel. A tool that quickly generates tracks based on various inputs will offer routing personnel the ability to analyze the parameter sensitivities and analysis of alternatives of these tracks more fully than is currently possible with manual methods.

Commander, Naval Meteorology and Oceanography Command is responsible for meeting the Navy's OTSR requirement. He has announced that the safe operation of maritime forces through the avoidance of severe weather still remains paramount. But with limited resources challenging the routing services and rising costs impacting naval operations, optimal routing will become increasingly more important. Due to this, the METOC community is under a mandate to operate as a fully efficient and effective enterprise. For the safety of maritime assets, this includes the consolidation and automation of OTSR services. Since 1993, Ocean Systems Incorporated has been producing the Vessel Optimization and Safety System and the Ship Tracking and Routing System software in order to assist commercial routing services by generating optimal ship voyage tracks based on ship and weather inputs (Ocean Systems Incorporated, 2005). These software packages were introduced to the METOC routing services in an attempt to assist in the development of route recommendations and diversions. However, routing personnel chose not to use the software due to the design favoring commercial merchant vessels and difficulty of use.

This thesis will build on the concept of using shortest path models to analyze ship routing (Brown, 1993) (Fagerholt, et al., 2000) (Lee, et al., 2002) and provide an operations research tool to aid in the planning for initial route recommendations and diversions. It will enable METOC personnel to provide decision makers multiple route recommendations quickly, based on a more complete and thorough analysis. Second, it will show that a time-tested proven algorithm for solving shortest paths used in conjunction with simple ship response functions can be easily automated and sufficient. The model for this thesis covers the Western Pacific (see Figure 1). We choose this theater of operation due to its political and operational importance.

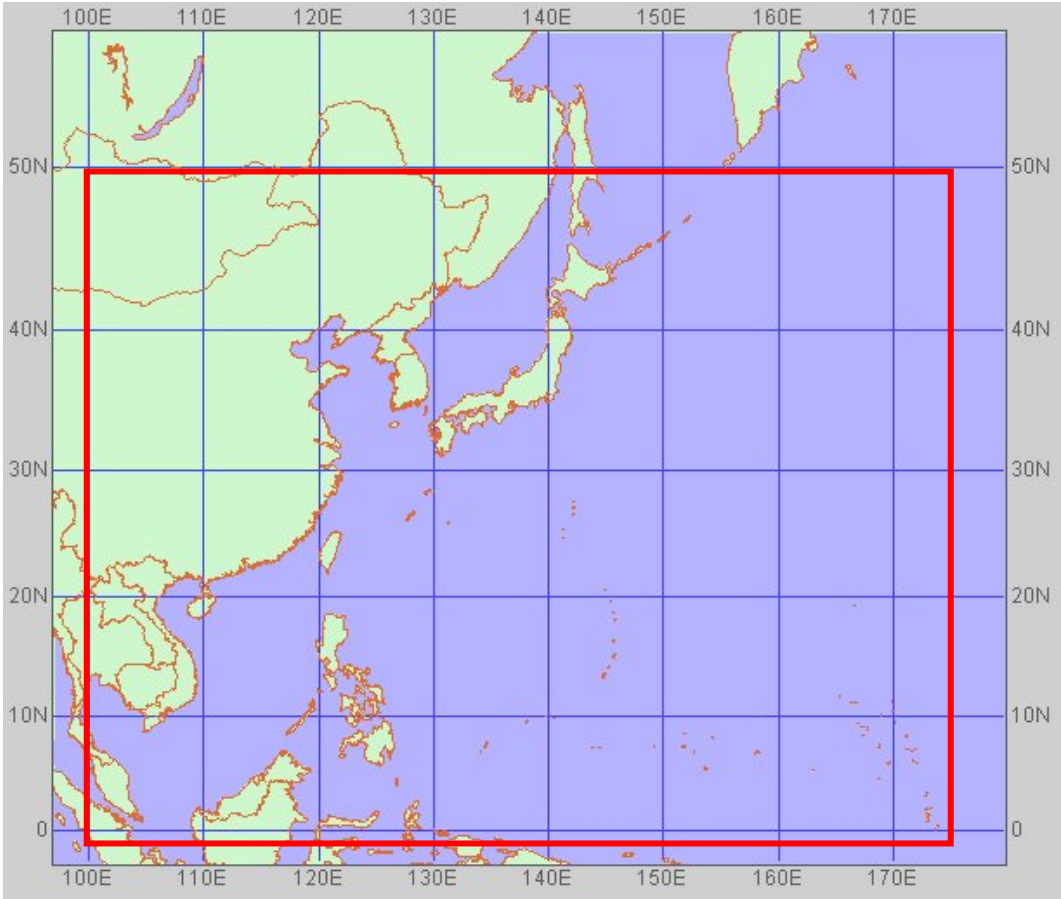


Figure 1. Model area coverage of Western Pacific.

Fleet Numerical Meteorology and Oceanography Center generates and distributes numerical weather prediction models, satellite imagery, and other weather related products and services. The atmospheric and ocean models used in this thesis are Navy Operational Global Atmospheric Prediction System (NOGAPS), Wave Watch 3 (WW3) (Naval METOC Command Instruction 3140.1L, 2000).

This thesis incorporates these numerical weather predictions into a shortest path network flow model in order to provide optimal tracks around adverse weather. The network is constructed of nodes and arcs overlying the Western Pacific Ocean region depicted in Figure 1. Nodes represent latitude and longitude positions, while the arcs connecting them represent navigable paths between nodes. Each node is populated with distance and time labels, wind and seas forecast predictions, and whether it is geographically accessible. It assumes the effects of currents, fog, and precipitation are

negligible. Arcs "costs" are defined as the nonnegative distances between nodes. We use a modified Dijkstra shortest path algorithm customized to solve these problems (Ahuja, et al., 1993, pg. 115). METOC personnel can manually add the latitude and longitude track output from this model to the Joint METOC Viewer (JMV) (Naval METOC Command Instruction 3140.1L, 2000) for visual viewing and further analysis, see Figure 2 for an example.

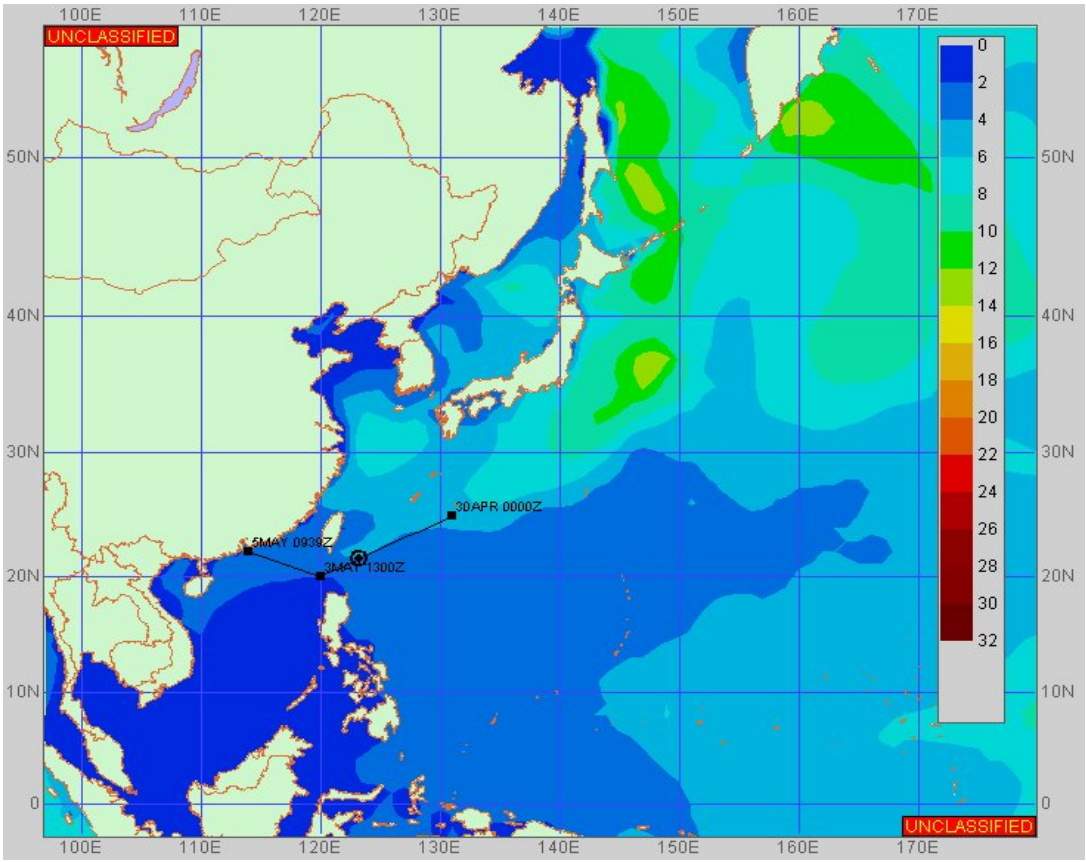


Figure 2. Screen shot of Western Pacific, WW3 Significant Wave Height data (feet), and ship track using Joint METOC Viewer software.

This thesis is a proof of concept to form the basis for possible testing in the future. It will not attempt to fully develop a fleet ready OTSR application, but rather demonstrate that this is a tool to analyze parameter sensitivities (speed, track, ship limits) when planning diverts as well as conducting analysis of alternatives. In order to objectively assess the model, test cases are run using recent model data. This will show if the model's diversion recommendations are safe routes by not exceeding wind and seas

limits. Then a comparison will be made between historical route diversions and the network model's diversion output using archived model data, as seen in Figure 3. This will show how well the model performs against divert tracks manually formulated by routing personnel.

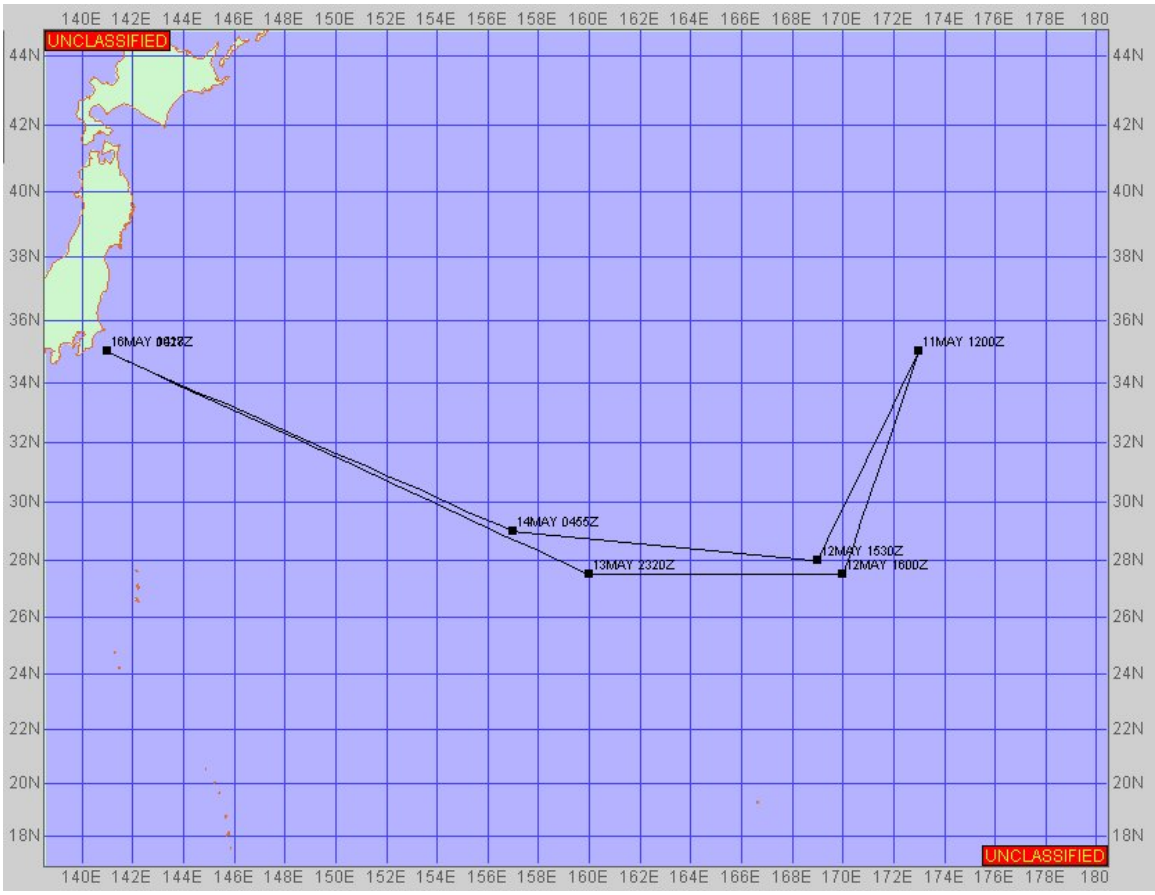


Figure 3. Notional example of divert route comparison.

THIS PAGE INTENTIONALLY LEFT BLANK

II. OPTIMUM TRACK SHIP ROUTING

Routing activities within the naval METOC community use several forms of formatted messages in order to perform OTSR effectively. This information is sent to the routing activity from the requesting unit using the U.S. Navy Movement Report system (Naval Warfare Publication 1-03.1, 1997) and the OTSR/Route Surveillance Request (Naval METOC Command Instruction 3140.1L, 2000).

The purpose of the movement report system is to gather and disseminate current location information to pertinent elements of the operational and administrative chain of commands regarding all U.S. Navy, Coast Guard, and Maritime Sealift Command (MSC) ships. Movement information is sent via a movement report message (see Appendix A). This message contains information including: ship class, point of departure, estimated time of departure, position of intended movement track and times, destination and time of arrival, intended and maximum speed of advance, and highest operational limits for head, beam, and following seas, and wind velocities. Units requesting OTSR need to provide this information to the routing activity at least 72 hours prior to departure (Naval METOC Command Instruction 3140.1L, 2000). While the movement report is acceptable, the unit requesting OTSR services may send a Route Surveillance Request (see Appendix B). Information contained in this message is more detailed, giving the routing activity better insight to the weather sensitivities and seaworthiness of the vessel. This information includes: loading characteristics, type of cargo loaded, deck loaded aircraft, material condition of ship which may affect seaworthiness, and operations scheduled enroute.

The routing activity analyzes this movement report and surveillance request information along with climatology, winds and seas forecasts, satellite data, and other METOC products in order to make an initial route recommendation that is provided to the requesting unit 36 to 48 hours prior to departure (Naval METOC Command Instruction 3140.1L, 2000). This recommendation provides the requesting unit with a route that minimizes the possibility of adverse weather during its transit. This is also the

beginning of surveillance where the requesting unit's progress is monitored for adverse weather avoidance and ends when the requesting unit reaches its destination.

Routing activities closely monitor elements of the ocean and atmosphere that may cause reduced speed, increased fuel consumption, ship damage, or personnel injury. Ocean and atmospheric elements of most concern are winds and seas. Ice and currents are considered under special transit circumstances, but rarely influence routing decisions. Winds and seas are the most important because a route can be considered optimal if the effects of winds and seas can be minimized. If wind and sea conditions are expected to exceed the ship's specified limits at the time of its departure, a delay in port recommendation will be made. Routing activities will adjust the departure time at this phase of the surveillance to avoid potentially hazardous weather as there are generally very few routing options close to coastal regions.

Once underway, the requesting unit will transmit a Daily OTSR Report to the routing activity. This report includes the vessel's position, course, speed, weather observation, and current time of arrival (see Appendix C). This information is crucial for the routing activity to properly monitor the vessel's transit progress. If the routing activity concludes that forecasted wind and sea conditions will exceed the ship's limit values during surveillance, a route diversion will be recommended (see Appendix D). A route diversion is an adjustment to the original transit track to avoid potentially hazardous weather and sea conditions forecasted to be encountered. While the divert track generally adds distance to the original track, this added cost is less expensive than the speed reduction, safety hazards, and added fuel consumption likely to occur while transiting adverse weather. However, certain weather systems may cause vessels to encounter a situation where time and maximum speed constraints do not allow a feasible divert solution. Concern for vessel safety and shiphandling now become the primary consideration over efficient timely transit. Therefore, routing activities may recommend either an adjustment in transit speed, where the vessel will increase or decrease speed in order to avoid the adverse weather with no change in track, or an evasion position in a general direction away from the adverse weather where the vessel can "wait out the storm". The routing activity will recommend a course and speed to regain original track

once the hazardous weather conditions have abated. OTSR services terminate once the requesting unit has transmitted its arrival notice upon entering port.

THIS PAGE INTENTIONALLY LEFT BLANK

III. NETWORK CONSTRUCTION

The network for this thesis covers the Western Pacific as shown in Figure 1. The area is bounded between latitude lines 0^o and 50^o North and longitude lines 100^o and 175^o East. The directed network $G = (N, A)$ is defined by a set N of n nodes and a set A of m directed arcs. Nodes are positioned at half-degree ($.5^o$) resolution along the lines of latitude and longitude in the defined area, including the boundaries. Figure 4 shows a two degree notional representation.

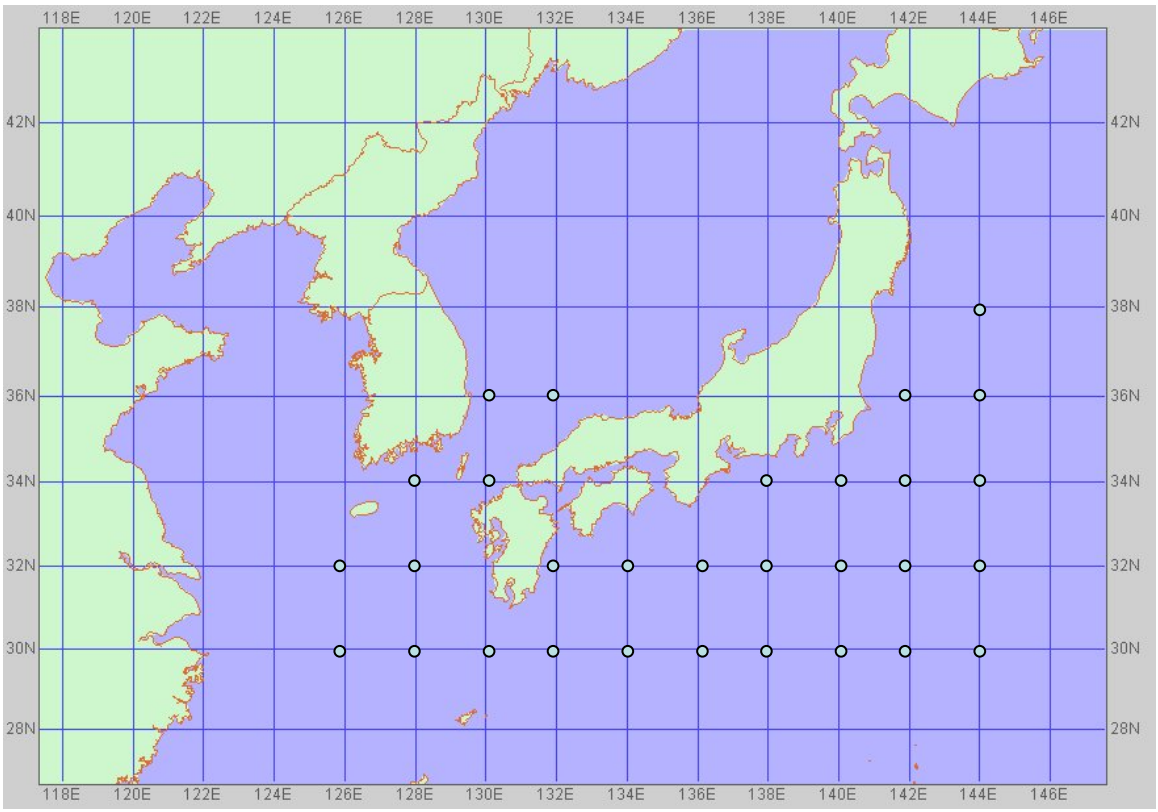


Figure 4. Notional representation of how network node positions correspond to latitude and longitude positions. 2-degree resolution here is for illustration only. The model uses half degree resolution.

Half-degree node resolution is chosen in order to match the half-degree resolution of the NOGAPS global weather model (Naval METOC Command Instruction 3140.1L, 2000). This results in 101 nodes on a line of longitude and 151 nodes on a line of latitude

for a total of 15,251 nodes. The corresponding latitude and longitude, in decimal degrees, for each node is as follows:

Node #	Latitude	Longitude
1	0.0	100.0
2	0.0	100.5
3	0.0	101.0
	.	
	.	
	.	
3321	10.5	174.5
3322	10.5	175.0
3323	11.0	100.0
3324	11.0	100.5
	.	
	.	
	.	
15251	50.0	175.0

For this network, node numbers are calculated given a latitude and longitude using the following formula:

$$N_o = (2\lambda)(151) + \left[2\left(\ell - 100\right)\right] + 1 .$$

Likewise, given a node number, the corresponding latitude and longitude can be calculated using the formulas:

$$\underline{\lambda} = \frac{(N_o - 1)/151}{2} \qquad \ell = \frac{\left[N_o - (2\lambda)(151) - 1\right] + 100}{2} .$$

where:

- N_o is the node number,
- λ is the latitude, and
- ℓ is the longitude.

These formulas are convenient for algorithm calculations and output. Nodes in the defined area that overlay land, shoal water, or are adjacent to small islands that intersect arcs between nodes are inaccessible. All other nodes over navigable water will be accessible. The graph is sparse; the number of arcs connecting neighboring nodes is far less than the number that would connect all node pairs.

Each arc $(i, j) \in A$ represents movement in one direction. Each arc has a corresponding "cost" that represents the directed distance in nautical miles between nodes *i* and *j* (see Figure 5).

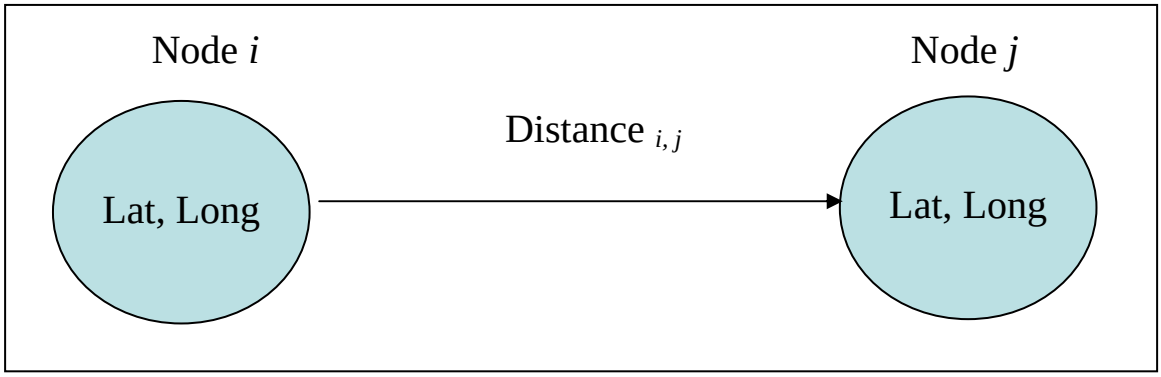


Figure 5. Node Legend.

Arc distances are pre-calculated using the great circle distance equation:

$$D = 60\left[a\cos\left(\sin(\lambda_1)\sin(\lambda_2) + \cos(\lambda_1)\cos(\lambda_2)\cos(\ell_1 - \ell_2)\right)\right].$$

where:

- D is the distance between geographic position 1 and geographic position 2,
- λ_1 is the latitude of geographic position 1,
- λ_2 is the latitude of geographic position 2,
- ℓ_1 is the longitude of geographic position 1, and
- ℓ_2 is the longitude of geographic position 2.

Arcs not only have a distance cost, but also represent the path in true bearing from node *i* to node *j*. Nodes that do not fall on the corners or boundary lines of the defined area will have an outdegree and indegree of eight. These eight arcs will connect node *i* to node *j* along the bearings 000^o, 045^o, 090^o, 135^o, 180^o, 225^o, 270^o, and 315^o true (see Figure 6).

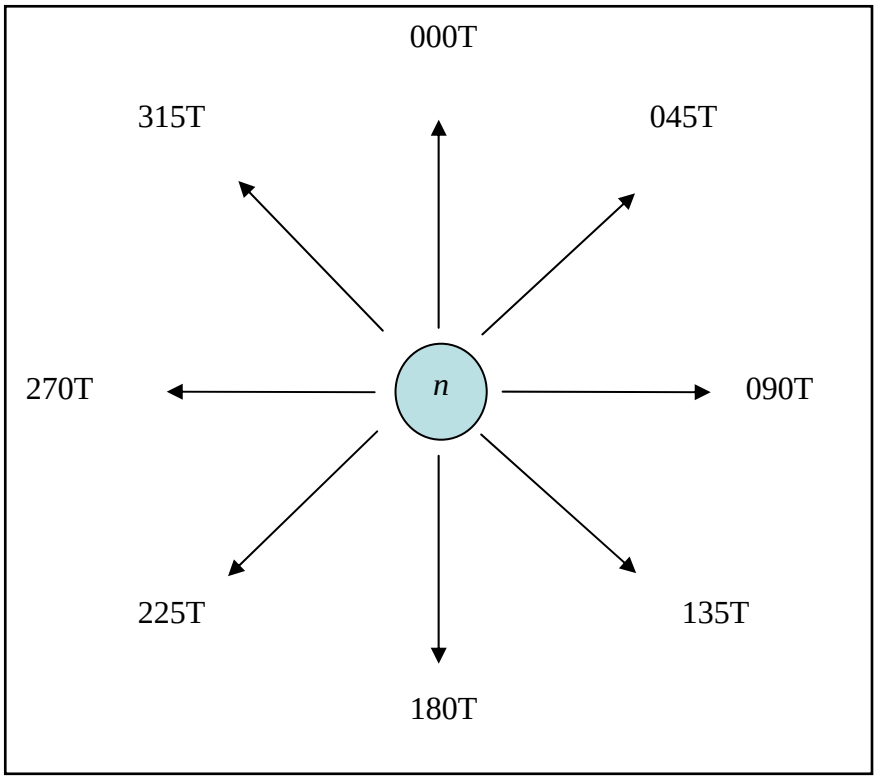


Figure 6. Spatial adjacencies with associated true bearings.

By restriction of the boundaries, corner nodes have an outdegree and indegree of three and boundary nodes, five. This results in the number of arcs m = 120,488. Any node with an outdegree of eight will be adjacent to nodes with node numbers:

$$N_o - 1, N_o + 1, N_o - 151, N_o + 151, N_o - 152, N_o + 152, N_o - 150, N_o + 150 .$$

Nodes on the corners and edges of the defined area will be a subset of the above node adjacencies depending on their position on the graph. Nodes and arcs for this network are stored and organized in a forward star representation, Appendix E (e.g., Ahuja, et al., 1993, p. 36). The forward star representation serves as an efficient data structure for cycling through the outgoing arcs of any node in the graph.

IV. ALGORITHM DEVELOPMENT

A. MODEL INPUT

Inputs to the model include ship characteristic data, movement report (movrep) data, and numerical weather prediction data as follows:

- The model will incorporate fuel consumption curves for 10 U.S. Navy ship classes (Schrady, et al., 1996) (Naval Sea Systems Command, 2005) (Appendix F);
- Speed reduction curves (National Geospatial Intelligence Agency, 2004) (Appendix G);
- Ship class;
- Ship wind and sea weather limits;
- Movement report speed, maximum allowable speed;
- Movement report track data, to include number of waypoints, latitude, and longitude; and
- NOGAPS wind speed and direction data, and WW3 sea heights and direction data. Weather forecast data is assumed to be 100% accurate.

An example of model input is shown in Appendix H.

B. DIJKSTRA'S ALGORITHM

A modified form of Dijkstra's algorithm (Appendix I), is used to find the "shortest" path from a source node *s* to a destination node *t*. "Shortest path" is defined in terms of shortest time, which is a function of distance (arc length) and ship speed:

$$transit\_time(i, j) = distance_{i,j} / speed\ .$$

For this model, we assume:

- The network is directed;
- The network contains a directed path from the source node *s* to every other node in the network;
- Arc lengths are integers; and
- All arc lengths are nonnegative. (Ahuja, et al., 1993, pp. 94-95).

Each node in the graph will have an associated distance label, time label, and predecessor label. Each node will also have corresponding weather data, and geographic accessibility data. At the beginning of the algorithm, each distance and time label are set equal to an arbitrarily large number. This number is set = *n*C, where *n* is the number of nodes, and C

is the largest arc length in the graph. The distance, time, and predecessor labels for the source node *s* are all set equal to 0. All calculations for ship's course, relative bearing aspect, and nearest forecast time are then calculated. From the source node *s*, all adjacent nodes in its forward star have their distance labels compared to an optimality condition defined as:

$$time_j > time_i + (transit\_time(i, j) + speed\_penalty(wx_{i,f}, a)) .$$

Where $speed\_penalty(wx_{i,f}, a)$ is a speed reduction function based on the wind speed and sea height forecast of node *i* at time *f* as well as the sea direction relative to the ship's hull aspect *a* (bow, beam, or stern). This function is based on the speed reduction curves as seen in Appendix G. We assume that the speed reduction curves apply to all classes of ships. A limit condition restricts our search to those nodes where the wind speed and sea heights at that node are within the ship's predefined weather limits. This limit condition is defined as: $ship\_limits_a < wx_{i,f}$. If this condition is met and there is an improvement in the time label through the optimality condition, then the time label, distance label, and the predecessor label are updated as follows:

$$time_j = time_i + (transit\_time(i, j) + speed\_penalty(wx_{i,f}, a)) ;$$
$$distance_j = distance_i + transit\_distance(i, j) ; \text{ and}$$
$$pred_j = i .$$

Where $transit\_distance(i, j)$ is the nonnegative geographic great circle arc distance from *node i* to *node j*. These conditions drive the algorithm to find the shortest time path. A binary heap function is used to "sort" and "sift" nodes and their corresponding time labels in order to find the $\min(d(j) : j \in N)$ and delete it from the heap, rather than the conventional Dijkstra method of permanently marking nodes. This reduces "big O" runtime from $O(n^2)$ to $O(mlog(n))$. The algorithm will repeat this cycle until the destination $node_t$ is reached. The least-time path is determined by stepping back through the predecessor array from node *t* to node *s*.

C. MODEL OUTPUT

Based on the input data and least-time path determined by Dijkstra's algorithm, the following is calculated:

- Distance, bearing, and time (hours) to transit between each movrep waypoint;
- Total movrep distance and transit time;
- Network least time “shortest” path in latitude/longitude positions;
- Distance of network path;
- Distance added to original movrep;
- Time to complete network path;
- Time added to original movrep;
- Fuel consumption for network path; and
- Monetary cost for fuel.

Appendix J depicts the output from the model given the inputs from section 4.1.

A time arrival calculation determines feasibility given by how many hours time early and/or late a vessel may arrive at its destination. If the time of arrival given the network path falls outside this allowable time arrival interval, the model calculates a recommended speed increase based on the model’s least-time path distance and the latest and earliest times, whichever is applicable. The user can then input the recommended speed and run the model. If the speed increase required is greater than the maximum allowable speed, then the model will prompt the user that other divert path options should be considered. Depending on the point in the transit timeline, these options include: delay in port, speed reduction with no change to track, or storm evasion.

THIS PAGE INTENTIONALLY LEFT BLANK

V. MODEL ANALYSIS

A. TEST CASES

We test the model's ability to route a vessel around adverse weather. We use four test cases using NOGAPS and WW3 12Z forecast data (Naval METOC Command Instruction 3140.1L, 2000) from May 5, 2005 and hypothetical ship tracks that purposely route through high sea states. Choice of ship class is arbitrary. Wind and sea limits are those most commonly recognized by ship captains and are not to be exceeded. Each case states the model inputs, model least-time track with time and distance calculations, computation time for model to find a least-time route, and Joint METOC Viewer display. The symbol d[i] is the distance traveled up to node *i*.

1. Test Case 1

Model inputs are:

Ship class – AO187;
Ship wind speed limits – 35 knots for bow, beam, and stern;
Ship sea heights limits – 12 feet for bow, beam, and stern;
Movement report speed – 12 knots;
Maximum allowable speed – N/A;
Number of waypoints, to include start and ending points – 2;
Movement report start position – 43.0N, 137.0E;
Movement report end position – 39.0N, 130.0E.

Model outputs are:

Total movement report distance is: 397.10 nm.
Total movement report transit time at 12.0 kts is: 33.09 hrs.

Model least-time track:

Node: 13061: d[i]:0.0 Latitude = 43.0N Longitude = 137.0E
Node: 12909: d[i]:37.84 Latitude = 42.5N Longitude = 136.5E
Node: 12757: d[i]:75.68 Latitude = 42.0N Longitude = 136.0E
Node: 12605: d[i]:113.52 Latitude = 41.5N Longitude = 135.5E
Node: 12453: d[i]:151.36 Latitude = 41.0N Longitude = 135.0E
Node: 12301: d[i]:189.2 Latitude = 40.5N Longitude = 134.5E
Node: 12149: d[i]:227.04 Latitude = 40.0N Longitude = 134.0E
Node: 11997: d[i]:264.88 Latitude = 39.5N Longitude = 133.5E
Node: 11845: d[i]:303.71 Latitude = 39.0N Longitude = 133.0E
Node: 11844: d[i]:328.36 Latitude = 39.0N Longitude = 132.5E
Node: 11843: d[i]:353.01 Latitude = 39.0N Longitude = 132.0E
Node: 11842: d[i]:377.66 Latitude = 39.0N Longitude = 131.5E

Node: 11841: d[i]:402.31 Latitude = 39.0N Longitude = 131.0E
Node: 11840: d[i]:426.96 Latitude = 39.0N Longitude = 130.5E
Node: 11839: d[i]:451.61 Latitude = 39.0N Longitude = 130.0E

Length of shortest path is approximately: 451.61 nm
Distance added to original movement report: 54.51 nm
Time to complete shortest path at 12.0 kts is: 37.63 hrs
Time added to original movement report: 4.54 hrs

Computation time: 62 milliseconds
Joint METOC Viewer Display (to reduce data clutter, wind speed is not displayed):

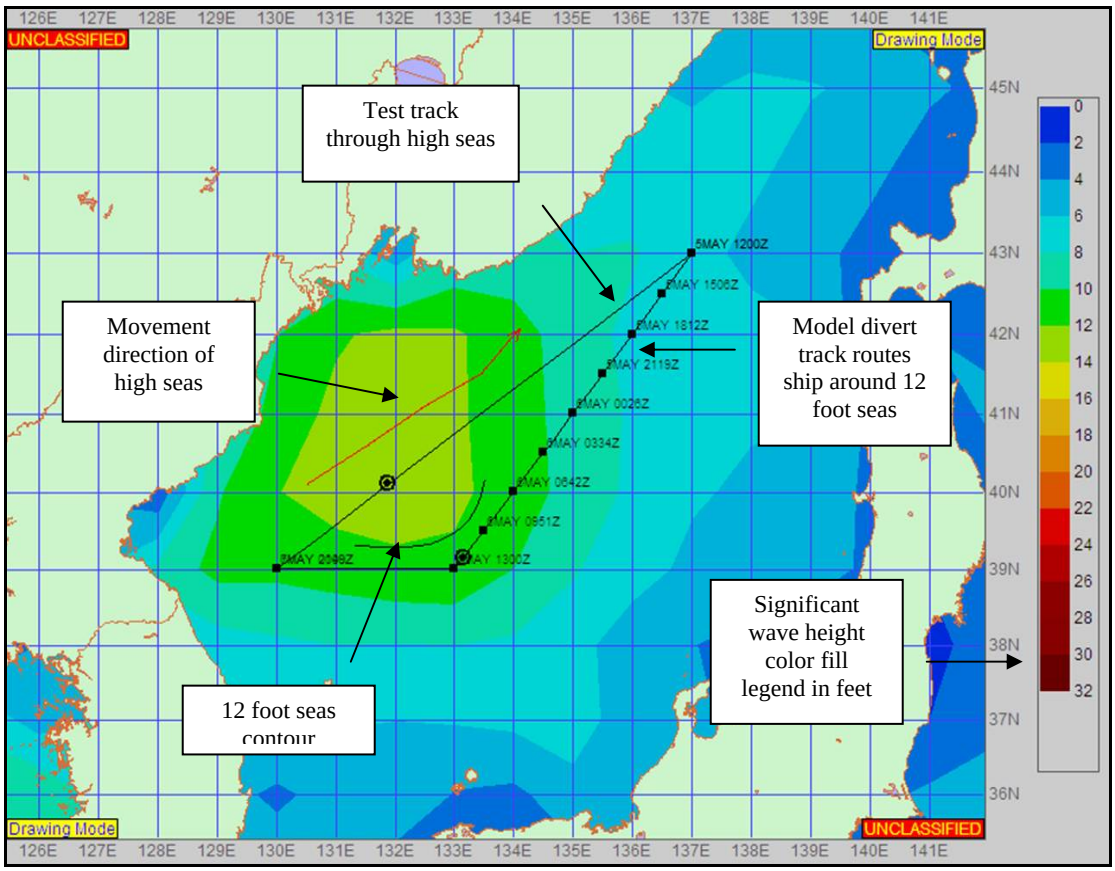


Figure 7. This screenshot captures the ship's position 24 hours into the transit of test case 1. The divert track routes the ship to the south and east avoiding 12 foot seas by approximately 20 nautical miles.

In test case 1, the ship approaches an adverse weather system in the Sea of Japan from the Northeast. This system builds the seas to a peak of 12-14 feet near the midpoint of the transit as it moves from the Southwest to Northeast. The model successfully

diverts the ship south and east of the high seas; avoiding the ship's 12 foot sea limits. This divert adds approximately 55 nautical miles and 4.5 hours to the original track.

2. Test Case 2

Model inputs are:

Ship class – AO187;
Ship wind speed limits – 35 knots for bow, beam, and stern;
Ship sea heights limits – 12 feet for bow, beam, and stern;
Movement report speed – 12 knots;
Maximum allowable speed – N/A;
Number of waypoints, to include start and ending points – 2;
Movement report start position – 35.0N, 174.0E;
Movement report end position – 33.0N, 162.0E.

Model outputs are:

Total movement report distance is: 608.06 nm.
Total movrep transit time at 12.0 kts is: 50.67 hrs.

Model least-time track:

Node: 10719: d[i]:0.0 Latitude = 35.0N Longitude = 174.0E
Node: 10407: d[i]:275.76 Latitude = 34.0N Longitude = 169.0E
Node: 10255: d[i]:315.49 Latitude = 33.5N Longitude = 168.5E
Node: 10103: d[i]:355.22 Latitude = 33.0N Longitude = 168.0E
Node: 9951: d[i]:394.95 Latitude = 32.5N Longitude = 167.5E
Node: 9799: d[i]:434.68 Latitude = 32.0N Longitude = 167.0E
Node: 9798: d[i]:460.73 Latitude = 32.0N Longitude = 166.5E
Node: 9797: d[i]:486.78 Latitude = 32.0N Longitude = 166.0E
Node: 9796: d[i]:512.83 Latitude = 32.0N Longitude = 165.5E
Node: 9795: d[i]:538.88 Latitude = 32.0N Longitude = 165.0E
Node: 9794: d[i]:564.93 Latitude = 32.0N Longitude = 164.5E
Node: 10091: d[i]:722.54 Latitude = 33.0N Longitude = 162.0E

Length of shortest path is approximately: 722.54 nm
Distance added to original movement report: 114.48 nm
Time to complete shortest path at 12.0 kts is 60.21 hrs
Time added to original movement report: 09.54 hrs

Computation time: 79 milliseconds

Joint METOC Viewer Display:

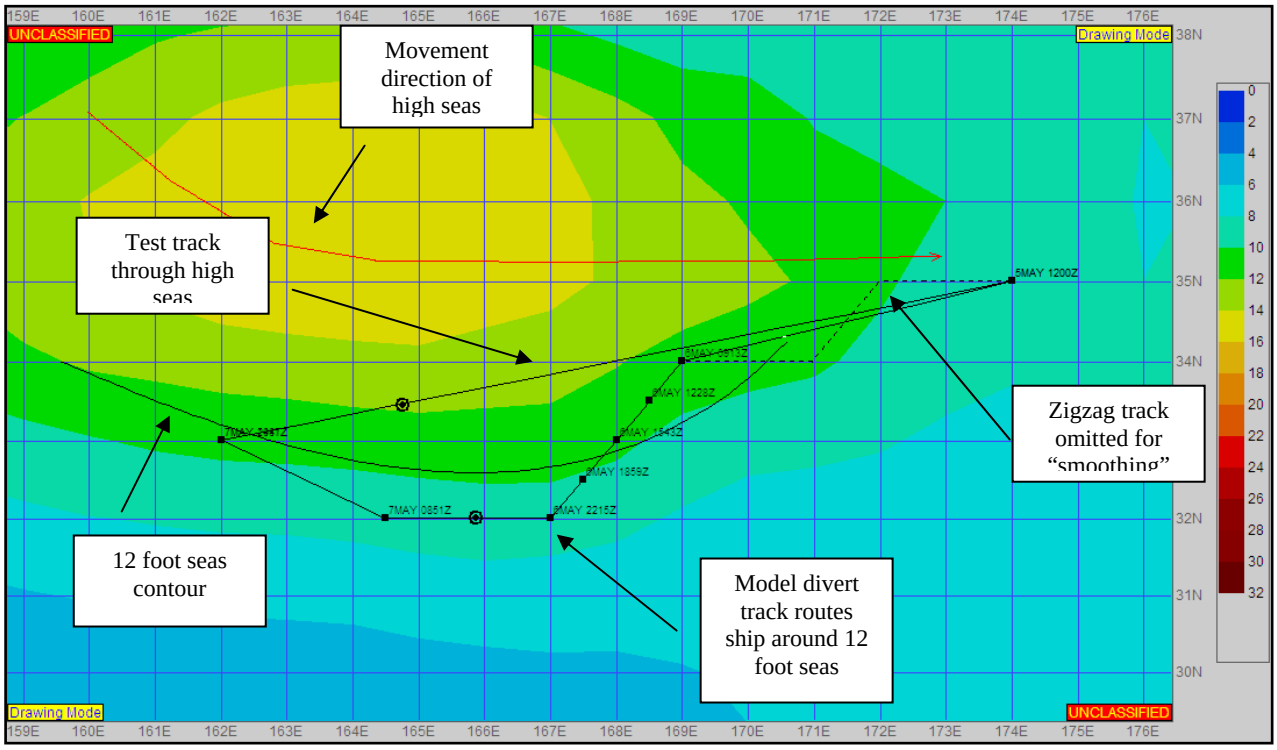


Figure 8. In this screenshot, the ship is 39 hours into its transit of test case 2. The model routes the ship south avoiding 12 foot seas by approximately 62 miles. Due to the model's node resolution, we omit nodes in order to "smooth" the divert track.

Case 2 tests the model ability to handle a large adverse weather system in the middle of the Western Pacific. This system originated off the east coast of Japan. Its movement begins to the Southeast then gradually curves to an easterly direction. Seas in the center of the system build to a peak of 16-18 feet. The test track takes the ship across the southern edge of the system where it exceeds its 12 foot sea limits. The model successfully diverts the ship to the south and avoids the 12 foot seas, however, some zigzagging occurs along the divert track. To correct this, some nodes in the model output were omitted in order to "smooth" the track. The zigzagging is caused by the ship's movement being restricted to model's nodal grid resolution. Refer back to Figures 4 and 6. This will cause the time and distances calculations to be slightly higher than the time and distance calculations for the track that appears in the display. When this occurs, the model output calculations can be considered an upper bound. All subsequent cases will have node omissions for "smoothing". The divert route adds approximately 114 nautical miles and 9.5 hours to the transit.

3. Test Case 3

Model inputs are:

Ship class – AO187;
Ship wind speed limits – 35 knots for bow, beam, and stern;
Ship sea heights limits – 10 feet for bow, beam, and stern;
Movement report speed – 13 knots;
Maximum allowable speed – N/A;
Number of waypoints, to include start and ending points – 2;
Movement report start position – 39.0N, 159.0E;
Movement report end position – 40.0N, 150.0E.

Model outputs are:

Total movement report distance is: 420.52 nm.
Total movrep transit time at 13.0 kts is: 32.35 hrs.

Model least-time track:

Node: 11897:	d[i]:0.0	Latitude = 39.0N	Longitude = 159.0E
Node: 12047:	d[i]:38.83	Latitude = 39.5N	Longitude = 158.5E
Node: 12197:	d[i]:77.66	Latitude = 40.0N	Longitude = 158.0E
Node: 12347:	d[i]:115.5	Latitude = 40.5N	Longitude = 157.5E
Node: 12497:	d[i]:153.34	Latitude = 41.0N	Longitude = 157.0E
Node: 12647:	d[i]:191.18	Latitude = 41.5N	Longitude = 156.5E
Node: 12495:	d[i]:229.02	Latitude = 41.0N	Longitude = 156.0E
Node: 12494:	d[i]:252.08	Latitude = 41.0N	Longitude = 155.5E
Node: 12493:	d[i]:275.14	Latitude = 41.0N	Longitude = 155.0E
Node: 12492:	d[i]:298.2	Latitude = 41.0N	Longitude = 154.5E
Node: 12491:	d[i]:321.26	Latitude = 41.0N	Longitude = 154.0E
Node: 12181:	d[i]:535.3	Latitude = 40.0N	Longitude = 150.0E

Length of shortest path is approximately: 535.3 nm
Distance added to original movrep: 0114.78 nm
Time to complete shortest path at 13.0 kts is 41.18 hrs
Time added to original movrep: 08.83 hrs
Computation time: 78 milliseconds

Joint METOC Viewer Display:

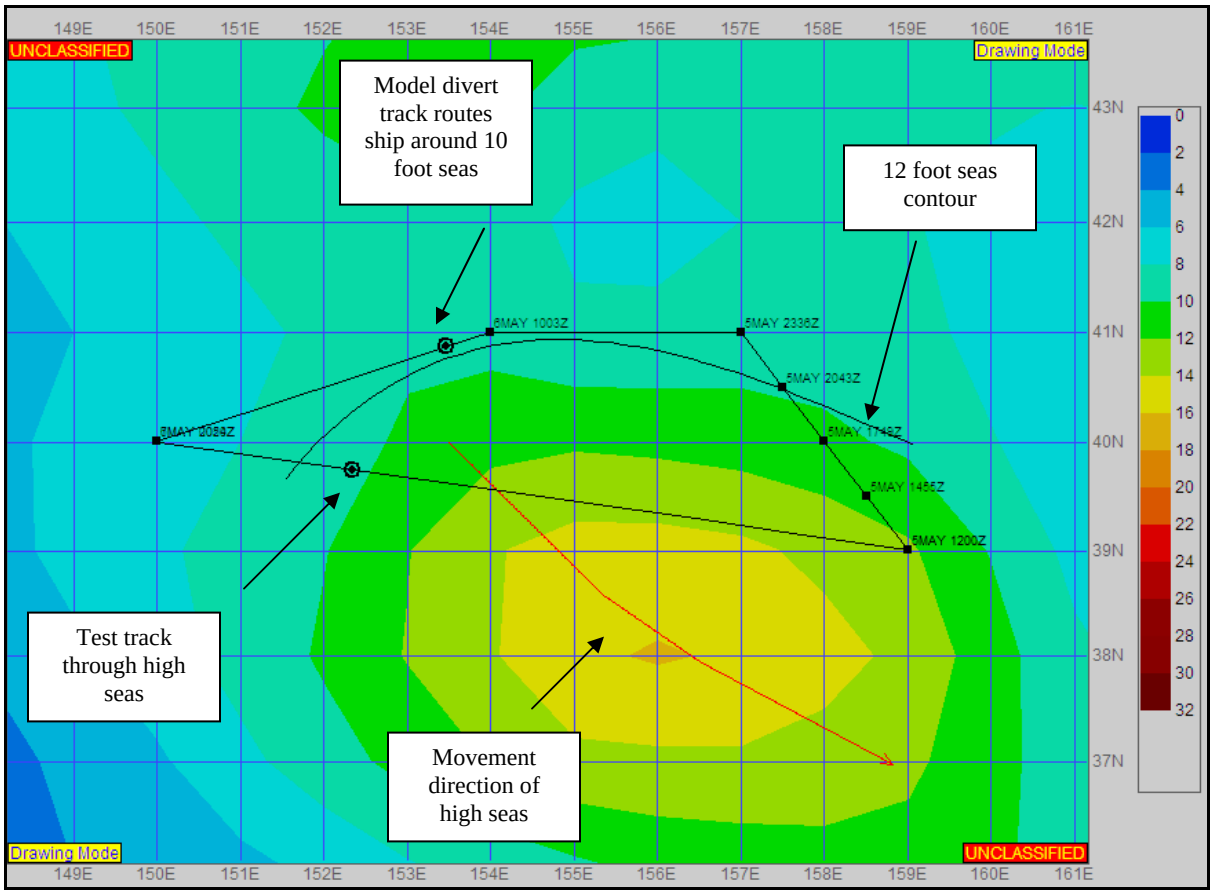


Figure 9. Here the vessel is at the 24 hour point in its transit of test case 3. The model routes the ship north approximately 14 nautical miles outside of 10 foot seas.

Test case 3 attempts to divert around the weather system previously described in test case 2. The test track differs as it takes the ship across the northern half of the system where seas reach the 12-14 foot range. This case also differs as the ship's speed is increased to 13 knots and sea limits are reduced to 10 feet. Again, the model successfully diverts the ship around high sea states with an added distance of approximately 115 nautical miles and added time of 9 hours.

4. Test Case 4

In this case we compare divert tracks with different sea limits. A test track takes the ship across 12-14 foot seas on the Northeast side of the weather system described in the two previous test cases. Fuel calculations are also added.

Model inputs are:

Track 1
Ship class – AO187;
Ship wind speed limits – 35 knots for bow, beam, and stern;
Ship sea heights limits – 12 feet for bow, beam, and stern;
Movement report speed – 13 knots;
Maximum allowable speed – N/A;
Number of waypoints, to include start and ending points – 2;
Movement report start position – 36.0N, 159.0E;
Movement report end position – 41.0N, 155.0E.

Track 2
Ship class – AO187;
Ship wind speed limits – 35 knots for bow, beam, and stern;
Ship sea heights limits – 10 feet for bow, beam, and stern;
Movement report speed – 13 knots;
Maximum allowable speed – N/A;
Number of waypoints, to include start and ending points – 2;
Movement report start position – 36.0N, 159.0E;
Movement report end position – 41.0N, 155.0E.

Track 1 model outputs are:

Total movement report distance is: 341.77 nm.
Total movement report transit time at 13.0 kts is: 26.29 hrs.

Model least-time track:

Node: 10991: d[i]:0.0 Latitude = 36.0N Longitude = 159.0E
Node: 11142: d[i]:30.06 Latitude = 36.5N Longitude = 159.0E
Node: 11594: d[i]:129.01 Latitude = 38.0N Longitude = 158.5E
Node: 11744: d[i]:167.84 Latitude = 38.5N Longitude = 158.0E
Node: 11894: d[i]:206.67 Latitude = 39.0N Longitude = 157.5E
Node: 12044: d[i]:245.5 Latitude = 39.5N Longitude = 157.0E
Node: 12194: d[i]:284.33 Latitude = 40.0N Longitude = 156.5E
Node: 12344: d[i]:322.17 Latitude = 40.5N Longitude = 156.0E
Node: 12494: d[i]:360.01 Latitude = 41.0N Longitude = 155.5E

Length of shortest path is approximately: 360.01 nm
Distance added to original movement report: 18.24 nm
Time to complete shortest path at 13.0 kts is 27.69 hrs
Time added to original movement report: 1.40 hrs

Computation time: 62 milliseconds

Additional fuel consumption for divert is approximately 638.34 gallons of F76.

Approximate monetary cost is $848.99.

Track 2 model outputs are:

Total movement report distance = 341.77 nm.
Total movement report transit time at 13.0 kts is: 26.29 hrs.

Model least-time track:

Node: 10991: d[i]:0.0 Latitude = 36.0N Longitude = 159.0E
Node: 11142: d[i]:30.06 Latitude = 36.5N Longitude = 159.0E
Node: 11293: d[i]:60.12 Latitude = 37.0N Longitude = 159.0E
Node: 11444: d[i]:90.18 Latitude = 37.5N Longitude = 159.0E
Node: 11595: d[i]:120.24 Latitude = 38.0N Longitude = 159.0E
Node: 11746: d[i]:150.3 Latitude = 38.5N Longitude = 159.0E
Node: 11896: d[i]:189.13 Latitude = 39.0N Longitude = 158.5E
Node: 12046: d[i]:227.96 Latitude = 39.5N Longitude = 158.0E
Node: 12196: d[i]:266.79 Latitude = 40.0N Longitude = 157.5E
Node: 12346: d[i]:304.63 Latitude = 40.5N Longitude = 157.0E
Node: 12496: d[i]:342.47 Latitude = 41.0N Longitude = 156.5E
Node: 12495: d[i]:365.53 Latitude = 41.0N Longitude = 156.0E
Node: 12494: d[i]:388.59 Latitude = 41.0N Longitude = 155.5E

Length of shortest path is approximately: 388.59 nm
Distance added to original movement report: 46.82 nm
Time to complete shortest path at 13.0 kts is 29.89 hrs
Time added to original movement report: 3.60 hrs

Computation time: 63 milliseconds

Additional fuel consumption for divert is approximately 1638.64 gallons of F76.
Approximate monetary cost is $2179.39.

Joint METOC Viewer Display:

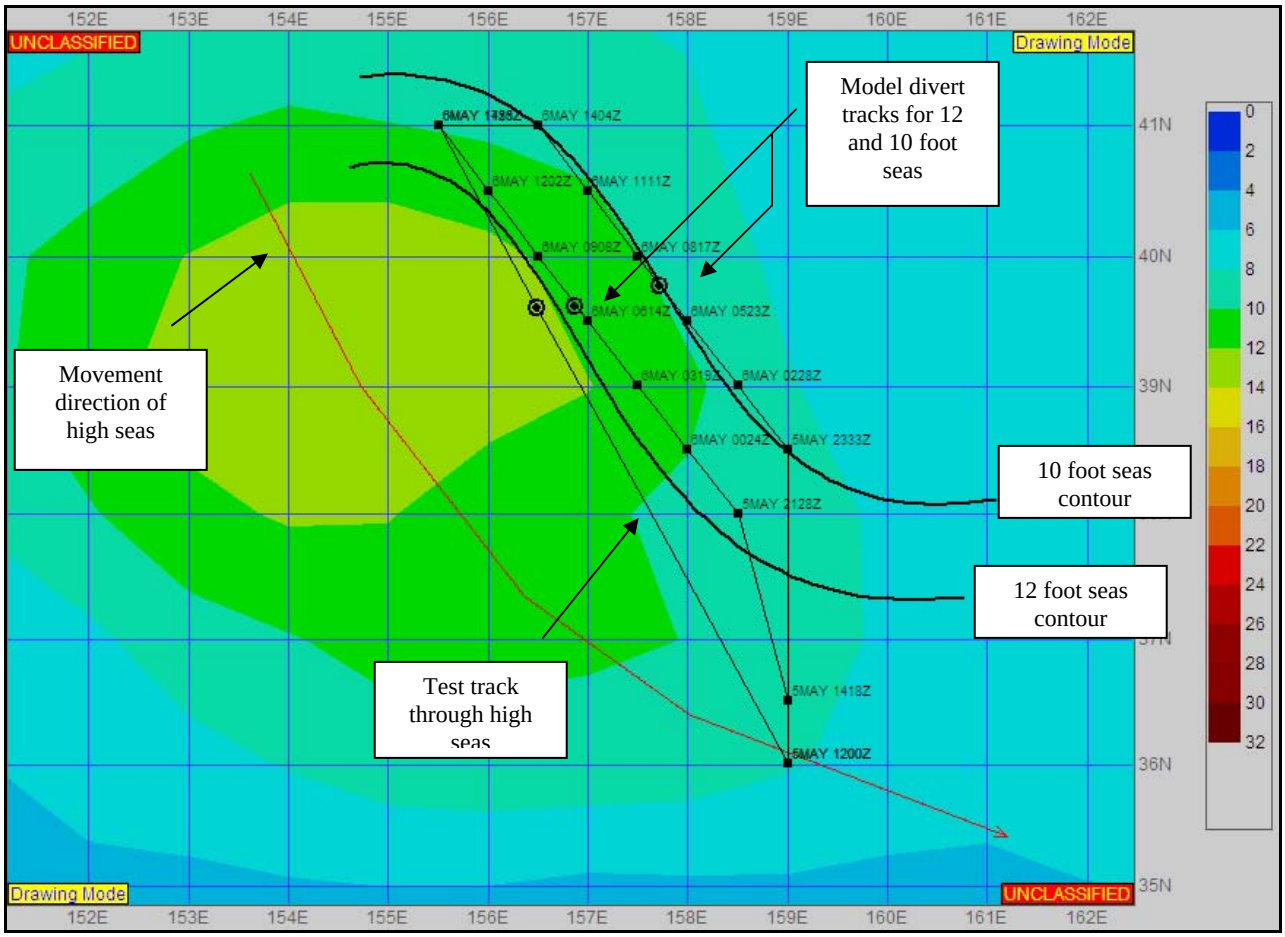


Figure 10. This screen shot is taken at the 19 hour point in the transit of test case 4. Each ship sails on the edge of its sea limits but neither exceeds their limits.

Each model-generated divert track avoids its respective sea limits.

In summary, this thesis has succeeded in matching a ship routing model with a Fleet Numerical product (Naval METOC Command Instruction 3140.1L, 2000). The network model quickly calculates optimal routes that avoid adverse weather showing that time and labor consuming manual routing techniques can be automated. Efficient analysis of alternatives is shown by changing input values for sea limits. Routing personnel can weigh the distance, time and fuel use differences against the benefits of avoiding different sea heights. Analysis of these results could assist routing personnel in identifying optimal routes more efficiently.

B. HISTORICAL CASE COMPARISON

In this section, we compare how the model performs against two real-world route diversions planned by ship routing personnel. Routing, divert, and prognostic reasoning data was collected from ship routing archive files at the Naval Pacific Meteorology and Oceanography Center, Pearl Harbor, HI. NOGAPS and WW3 archived model data is from Fleet Numerical Meteorology and Oceanography Center. Model data used is at a one degree resolution vice half degree. This is due to data space conservation efforts in the archiving of model data. The archived model data is also in format that is not compatible with the Joint METOC Viewer software, therefore not allowing visual representation of weather parameters as in the test cases. This may cause some loss of accuracy in the model divert results. Also, routing personnel take into account far more data sources then our model, therefore, the prognostic reasoning for their diverts will be considered in analyzing the model. For brevity, only model divert results and Joint METOC Viewer Display will be shown.

1. Historical Case 1

This case concerns a vessel transiting from Guam to Tokyo Wan in January, 2005, at a speed of 8.5 knots and wind and sea limits of 35 knots and 15 feet respectively. NOGAPS and WW3 forecast data from January 1, 2005, 12Z run is used. Model output is as follows:

Total movement report distance is: 1335.77 nm.
Total movement report transit time at 8.5 kts is: 157.15 hrs.

Length of manual divert: 1404.20 nm
Distance added to original movement report: 68.43 nm
Time added to original movement report: 8.05 hrs

Length of shortest path using 15 foot beam limits is approximately: 1386.70 nm
Distance added to original movement report: 50.93 nm
Time added to original movement report: 5.99 hrs

Length of shortest path using 12 foot beam limits is approximately: 1449.11 nm
Distance added to original movement report: 113.34 nm
Time added to original movement report: 13.33 hrs

From the routing activity's prognostic reasoning, and as seen in Figure 11, there is a 996 millibar low pressure system over Northern Japan moving in a northeasterly

direction. The low pressure system continues to intensify during the ship's transit. A 1038 millibar high pressure system is located over Korea. The interaction between these systems produced high winds and seas from the northwest. Weather conditions were then forecasted to improve as the ship approached 32 degrees North latitude. Routing personnel chose this divert in order to prevent the ship from exceeding its limits and to put the winds and seas on the ship's bow vice beam to reduce rolling motion. The model divert track using 15 foot sea limits for the bow, beam, and stern roughly correlates to the manual divert. It diverts the ship on a northerly heading as conditions improve. This divert saves approximately 18 nautical miles and 2 hours. Our model can take into account ship motion preference. By changing the ship's beam limits to 12 feet vice 15, the model divert track continues to the northwest then to the coastal approaches of Tokyo Wan. Next, we use the legacy manual divert vice the ship's movement report in the model in order to see whether or not the ship's limits are exceeded. Figure 12 provides a closer view showing that the model's track correlates very closely to the legacy manual divert track.

In summary, the first historical comparison reaffirms the automation possibilities for OTSR and serves as a good example of other cases that will appear in the future. Taking a ship's sea keeping capability into consideration is often done when planning OTSR divert tracks. By changing ship input characteristics and rerunning the model, ship routers can quickly analyze divert track metrics that take better sea keeping into account against those that do not.

Joint METOC Viewer Display:

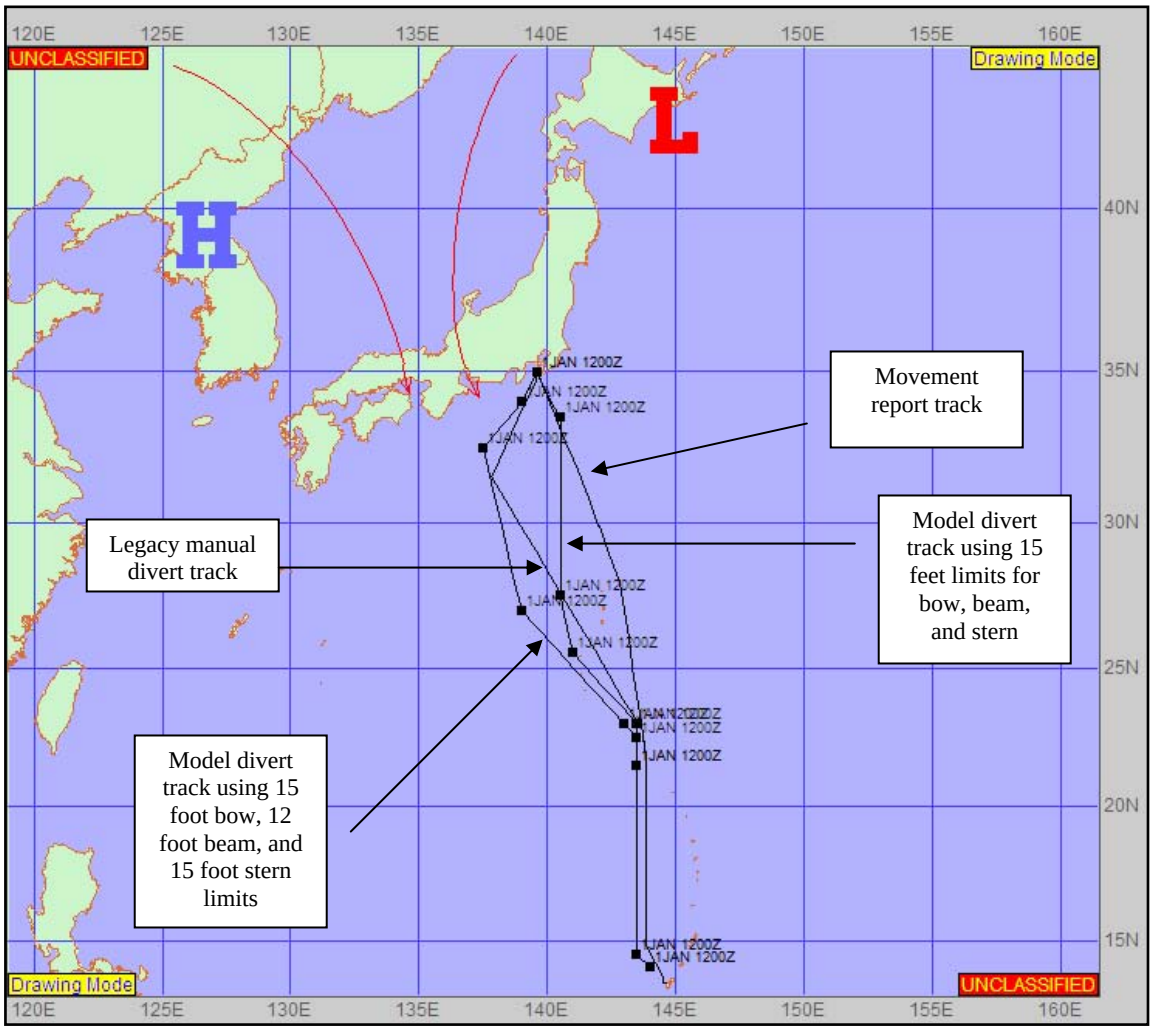


Figure 11. In this case, the model divert tracks correlate with the legacy manual divert track. Routing personnel chose to continue the ship's transit on a northwesterly heading to keep the winds and seas on the bow vice beam. The model can take this into account. When beam limits are reduced to 12 vice 15 feet, the model continues the ship on a more northwesterly heading than the model divert with 15 foot limits for the bow, beam, and stern.

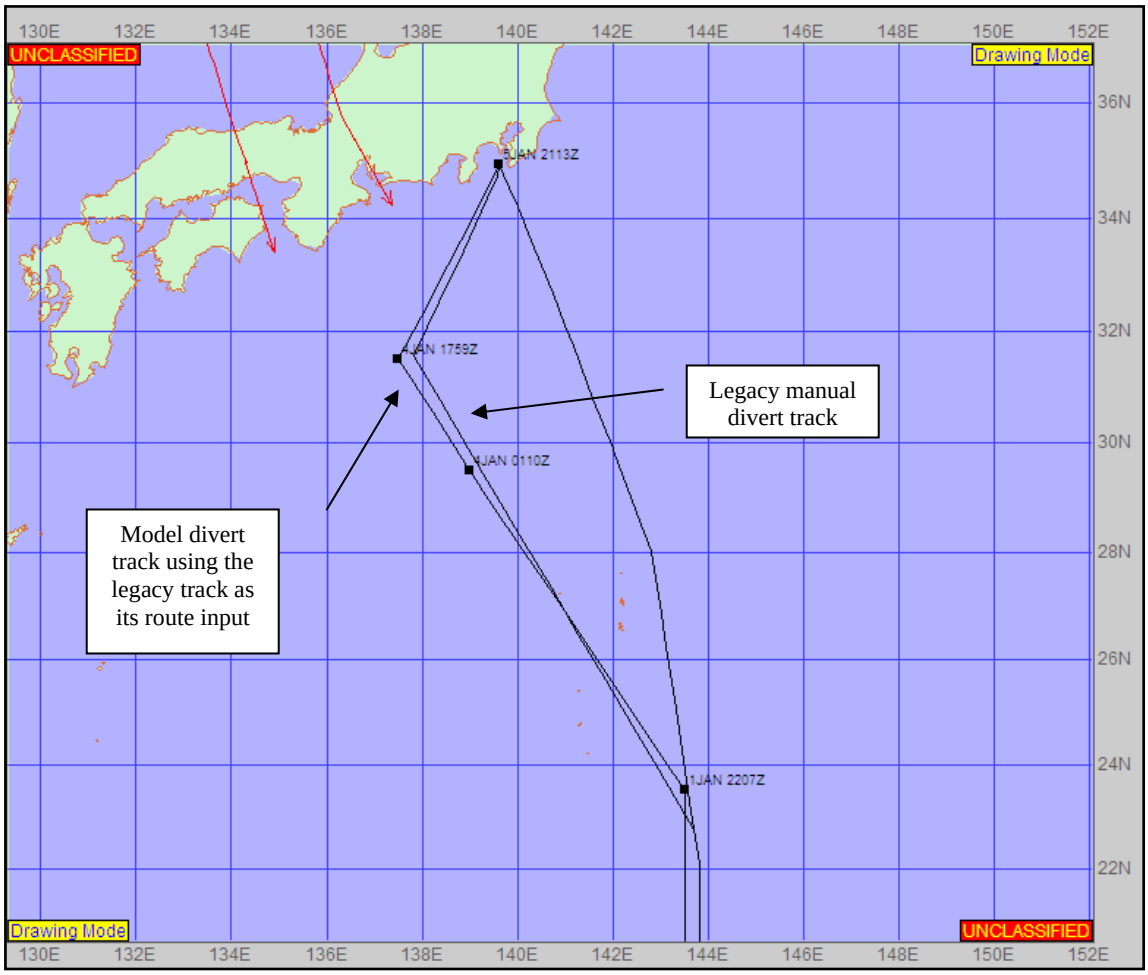


Figure 12. In a closer view, we see that the model divert track and legacy manual divert track matches closely. This confirms that the legacy track did not exceed the ship's limits.

2. Historical Case 2

This is a generalized case involving a vessel transiting from the International Date Line to Tokyo Wan in April, 2005, at a speed of 9 knots and with wind and sea limits of 35 knots and 15 feet respectively. NOGAPS and WW3 forecast data from April 27, 2005, 00Z run is used. Model output is as follows:

Total movement report distance is: 1101.39 nm.
Total movement report transit time at 9.0 kts is: 122.37 hrs.

Length of original divert: 1153.23 nm
Distance added to original movement report: 51.84 nm
Time added to original movement report: 5.76 hrs

Length of shortest path is approximately: 1157.56 nm

Distance added to original movement report: 56.17 nm
Time added to original movement report: 6.24 hrs

From the routing activity's prognostic reasoning, and as seen in Figure 13, Typhoon Sonca was moving northeast at 14 knots and continued in that direction during the ship's transit. It was forecasted to weaken as it moved north. Routing personnel recommended the ship sail on a divert track slightly south in order to get behind the storm and avoid its remnants. Tropical storms never curve towards the equator; therefore it is prudent to do this. The model divert corresponds closely to the legacy manual track and demonstrates the model's potential to incorporate more advanced tropical storm forecasts in the future.

Joint METOC Viewer Display:

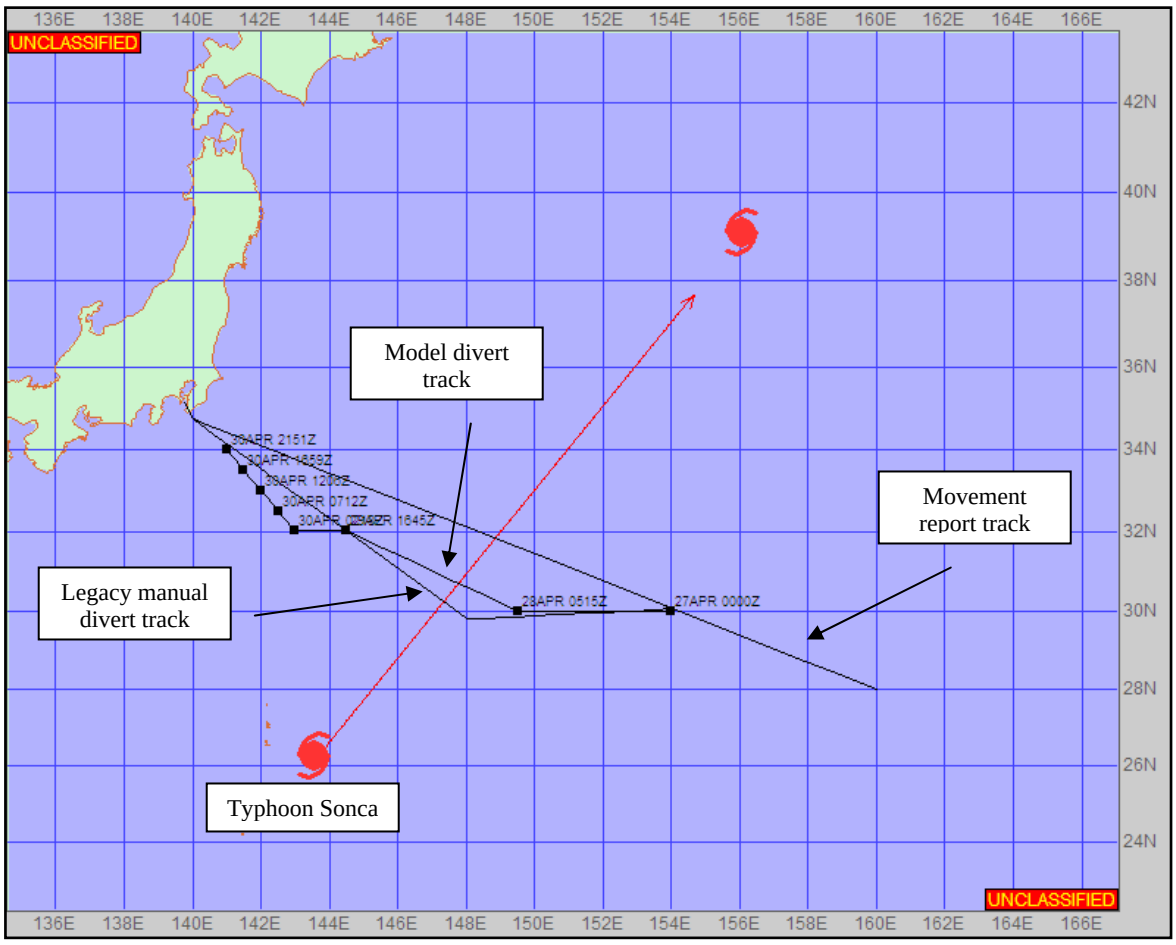


Figure 13. Here the model divert track correlates closely to the manual legacy divert.

VI. CONCLUSION AND FUTURE WORK

A. CONCLUSION

Optimum Track Ship Routing is the primary maritime support service performed by the METOC Command. OTSR personnel perform this service effectively by issuing initial route recommendations, advisories, and diversion routes. These services minimize the effects of winds and seas, but come at a cost in time and labor. Ship routing companies have developed advanced software packages to aid in the formulation of optimum shipping routes; however, METOC ship routers are reluctant to use this complex software.

In this thesis, we have devised a network ship routing model that solves optimal path problems using a modified version of Dijkstra's shortest path algorithm and a basic ship response function. We have established the model's fidelity using test cases. As shown, the model avoids adverse weather and solves the least-time path to a destination. It calculates useful time, distance, and fuel consumption metrics in order to quantify routing decisions. The model also demonstrates that manual routing techniques involving numerous calculations and chart plotting can be automated and solutions generated in milliseconds. We have identified how the results could be used by ship routing personnel to assist in analyzing alternatives and aiding ship routing decisions. The model's performance against the historical cases reaffirms the model's skill and gives insight to improvements that can be made in the future.

B. FUTURE WORK

The optimum routing demonstrated by this thesis is only a beginning. The model incorporates only one of the many products produced by the METOC command. There are numerous opportunities for future work, they include:

- Modifying the network structure by increasing node resolution and the number of spatial adjacencies. This will give more direct routes and increase route calculation accuracy. They model area coverage can also be expanded to cover other naval operating areas;
- Incorporating different model forecast input. This will allow routing personnel to analyze routes using data from different models and then choose the model that is currently performing better;

- Automating input and output functions;
- Compatibility with existing METOC products. The model is written in Java code (Savitch, 2001), allowing it to be easily transferred to other systems (Appendix K); and
- Compiling outputs into various ensembles for rapid visual analysis.

APPENDIX A. SAMPLE MOVEMENT REPORT

IMMEDIATE

O P 301700Z DEC 04 PSN 538836H33

FM USNS RICHARD G MATTHIESEN

TO AIG 55

INFO COMSCPAC SAN DIEGO CA
COMSC WASHINGTON DC//PM51//
MSCREP SAN FRANCISCO BAY CA
DESC FT BELVOIR VA//DESC-OII//
DESC PACIFIC CP H M SMITH HI//SAPO-H//
DESC PACIFIC CP H M SMITH HI//SAPO-H//
FLENUMMETOCCEN DATA MONTEREY CA//DATA//
NAVPACMETOCCEN PEARL HARBOR HI//30//
CNO WASHINGTON DC//N096/N311XW//
COMTHIRDFLT
COMNAVREG SW SAN DIEGO CA
MSCREP SEATTLE WA//N3//
COMSCPAC SAN DIEGO CA//N3//
COMNAVREG NW SEATTLE WA
COMCOGARD LALB LOS ANGELES CA
NAVPACMETOCCEN YOKOSUKA JA//30//
USNS RICHARD G MATTHIESEN//N67//
MRC NORFOLK VA//MRC/311//
AMVER CENTER MARTINSBURG WV//TLX414//
FLENUMMETOCCEN DATA MONTEREY CA//30//
ONI IFE WASHINGTON DC//JJJ//

BT
UNCLAS //N03123//
OOO MOVREP USNS RICHARD G MATTHIESEN, NBBP, OTSR/WEAX 01//
POS L 43-00N7 125-44W6, 301600Z7 DEC RH 011.0K2 02//
VIA L 40-00N4 125-35W6, 310900Z3 DEC 03//
VIA L 38-00N1 124-30W0, 312100Z7 DEC 04//
VIA L 33-40N0 121-27W3, 020030Z5 JAN 05//
VIA L 33-38N7 120-10W4, 020630Z1 JAN 06//
VIA L 33-36N5 118-15W6, 021530Z1 JAN 07//
VIA L 33-41N1 118-11W2, 021600Z9 JAN 08//
ETA P LONG BEACH CA, 021600Z9 JAN 09//
SUP BUNKERS IFO 6934 BBLS 90 PCT/MGO 1404 BBLS 92 PCT 10//
SUP SLOPS ENG 18 BBLS/3 PCT 0 BBLS/00 PCT DECK 11//
SUP FOR OTSR VSL IS LADEN MAX SEA/WIND HEAD 20/40 12//
SUP FOR OTSR BEAM 20/40 FOLL 30/40 13//

BT

Requesting unit

Action and info addees

Message header that indicates OTSR services are requested.

Latitude and longitude positions and times that constitute the movement report track.

Speed of transit

Estimated time of arrival

Supplemental OTSR comments that includes the ship's loading status and wind and sea limits

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B. SAMPLE OTSR ROUTE REQUEST

FM USS SHOUP ← Requesting unit

TO NAVPACMETOCCEN SAN DIEGO CA

INFO FLENUMMETOCCEN DATA MONTEREY CA
NAVPACMETOCCEN YOKOSUKA JA
NAVPACMETOCCEN PEARL HARBOR HI

UNCLAS //N03145//

MSGID/GENADMIN/SHOUP//
SUBJ/OTSR ROUTE REQUEST//
REF/A/DOC/NAVMETOCCOM 3140.1L/15SEP2000//
AMPN/UNITED STATES NAVY METEOROLOGICAL AND OCEANOGRAPHIC SUPPORT SYSTEM MANUAL.//

RMKS/1. SHOUP (DDG).
2. (U) APRA HARBOR GU, 180100Z0 DEC 04.
3. VICTORIA HARBOR HK, ETA 240045Z5 DEC 04.
4. 13 KTS, MAX ACCEPTABLE SOA 22 KTS.
5. DRAFT: 32 FT.
6. HIGHEST OPERATIONAL LIMITS: HEAD 18 FT, BEAM 15 FT, FOLLOWING 20FT, WIND 35 KTS.
7. FLIGHT OPS TO BE CONDUCTED UNDERWAY. UNREP TO BE SCHEDULED ON 22DEC04. DETAILS TO BE INCLUDED IN DAILY OTSR REPORT.

8. N/A.//

BT

OTSR information that includes:

1. Vessel name and class.
2. Departure port and time of departure.
3. Arrival port and estimated time of arrival.
4. Transit speed and maximum speed of advance.
5. Vessel's draft.
6. Highest operational weather limits.
7. Operations to be conducted while in transit.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX C. DAILY OTSR REPORT

IMMEDIATE

O R 301600Z DEC 04 PSN 538119H27

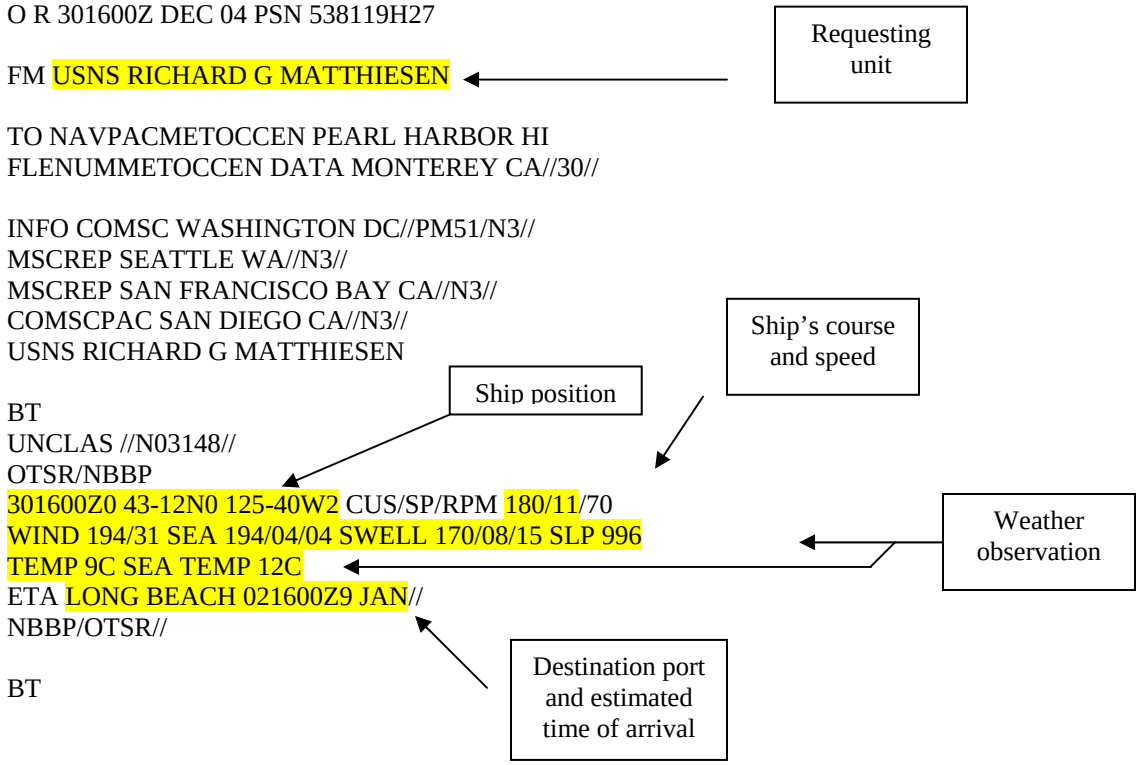
FM USNS RICHARD G MATTHIESEN

TO NAVPACMETOCCEN PEARL HARBOR HI
FLENUMMETOCCEN DATA MONTEREY CA//30//

INFO COMSC WASHINGTON DC//PM51/N3//
MSCREP SEATTLE WA//N3//
MSCREP SAN FRANCISCO BAY CA//N3//
COMSCPAC SAN DIEGO CA//N3//
USNS RICHARD G MATTHIESEN

BT
UNCLAS //N03148//
OTSR/NBBP
301600Z0 43-12N0 125-40W2 CUS/SP/RPM 180/11/70
WIND 194/31 SEA 194/04/04 SWELL 170/08/15 SLP 996
TEMP 9C SEA TEMP 12C
ETA LONG BEACH 021600Z9 JAN//
NBBP/OTSR//

BT



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX D. DIVERT EXAMPLE

FM NAVPACMETOCCEN PEARL HARBOR HI

METOC routing activity

TO USCGC RUSH
CNO WASHINGTON DC(uc)
CDR USPACOM HONOLULU HI(uc)
COMPACFLT PEARL HARBOR HI
COMTHIRDFLT
COMPACAREA COGARD ALAMEDA CA
CCGDSEVENTEEN JUNEAU AK
NAVPACMETOCCEN PEARL HARBOR HI(uc)
NAVLANTMETOCCEN NORFOLK VA(uc)

Unit being advised to divert from present route.

UNCLAS //N03145//

MSGID/GENADMIN/NAVPACMETOCCEN PEARL HARBOR HI//
REF/A/MOVREP/RUSH/142107ZFEB2005/-/NOTAL//
REF/B/TEL/USCGC RUSH OPS/141240ZFEB2005//
NARR/REF B, PHONECON BETWEEN OTSRDO PH AND OPSO USCGC RUSH.//
POC/NPMOC/CDO/NAVPACMETOCCEN/-/TEL:COMM (808)474-4834
/SECTEL:DSN (315)474-4834/EMAIL:CDO.NPMOC_JTWC@NAVY.MIL//
AKNLDG/YES/INST:REQ ACKNLG RECEIPT AND ADVISE INTENTIONS TO ORIG//
RMKS/1. IVO REF A AND AS DISCUSSED REF B, OTSR RECOMMENDS IMMEDIATE DIVERT FROM CURRENT POSITION AND RH TO 45.0N9 147.0W2 AND INCREASE SOA TO 15 KTS AS SAFE NAVIGATION PERMITS THEN RETURN TO ORIGINAL SOA OF 12 KTS RH TO KODIAK, AK. THIS DIVERT WILL ADD APPROX 74 NM TO ORIGINAL TRACK REF A.

Divert route

2. AT 15/00Z A 969 MB LOW IN THE CENTRAL PACIFIC WILL MOVE EAST-NORTHEAST TO NR 46.0N0 155.0W1 AND INTERACT WITH HIGH TO THE EAST OF YOUR CURRENT TRACK PRODUCING 35-40 KT WINDS AND WESTERLY SEAS 30-35 FT.
3. RECOMMENDED DIVERT WILL AVOID SHIPS EXPOSURE TO 30-35 FT SEAS AND ADVANCE YOUR ETA BY OVER 22 HRS REF A.
4. ALONG RECOMMENDED DIVERT TRACK EXPECT NORTHEAST WINDS 10-15 KTS BACKING TO SOUTH 15-20 KTS BY 16/12Z. SEAS WILL BE WEST-NORTHWEST 11-15 FT, BUILDING TO 15-18 FT BY 16/12Z THEN GRADUALLY ABATING TO 11-15 FT BY 18/06Z.
5. OTSR WILL CONTINUE TO MONITOR YOUR UNIT AND WILL ISSUE FURTHER ADVISORIES/RECOMMENDATIONS AS CONDTIONS WARRANT.

Weather forecast and prognostic reasoning

6. REQUEST SUBMIT THREE HOURLY WEATHER OBSERVATIONS IMMEDIATE PRECEDENCE TO "NAVPACMETOCCEN PEARL HARBOR HI" FOR WINDS 35 KTS AND GREATER OR SEAS 12 FT AND GREATER.
7. OTSRDO: BALLUCANAG/ALFELOR.//

Optimum Track Ship Routing Duty Officers

BT

THIS PAGE INTENTIONALLY LEFT BLANK

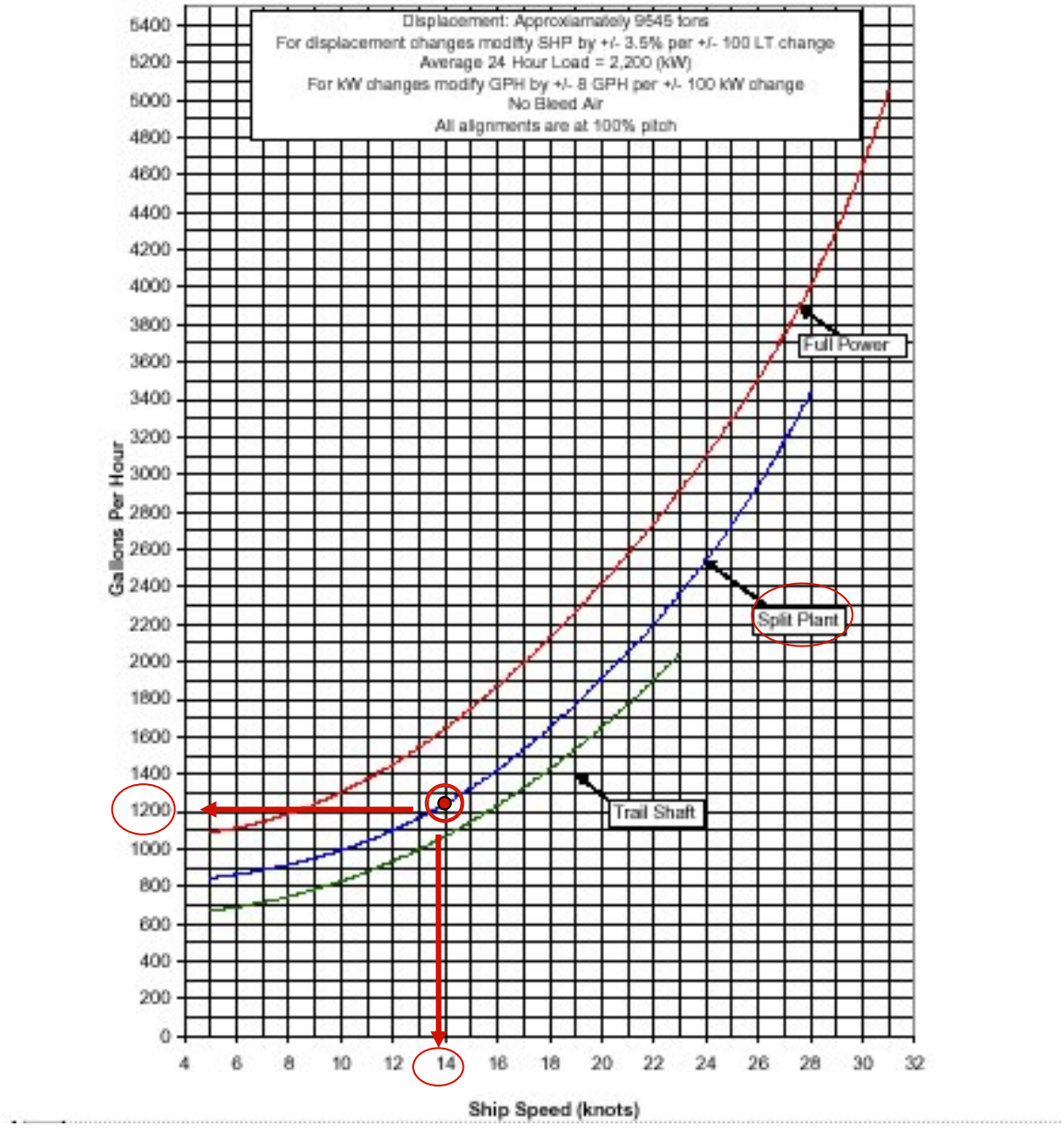
APPENDIX E. EXAMPLE OF FORWARD STAR DATA STRUCTURE REPRESENTATION

node	point
1	1
2	2
3	4
4	6
n+1	m+1

arc	tail	head	distance
1	1	2	20
2	2	3	15
3	2	4	12
4	3	2	35
5	3	4	25
6	4	2	11
7	4	3	14
m	m	m	m

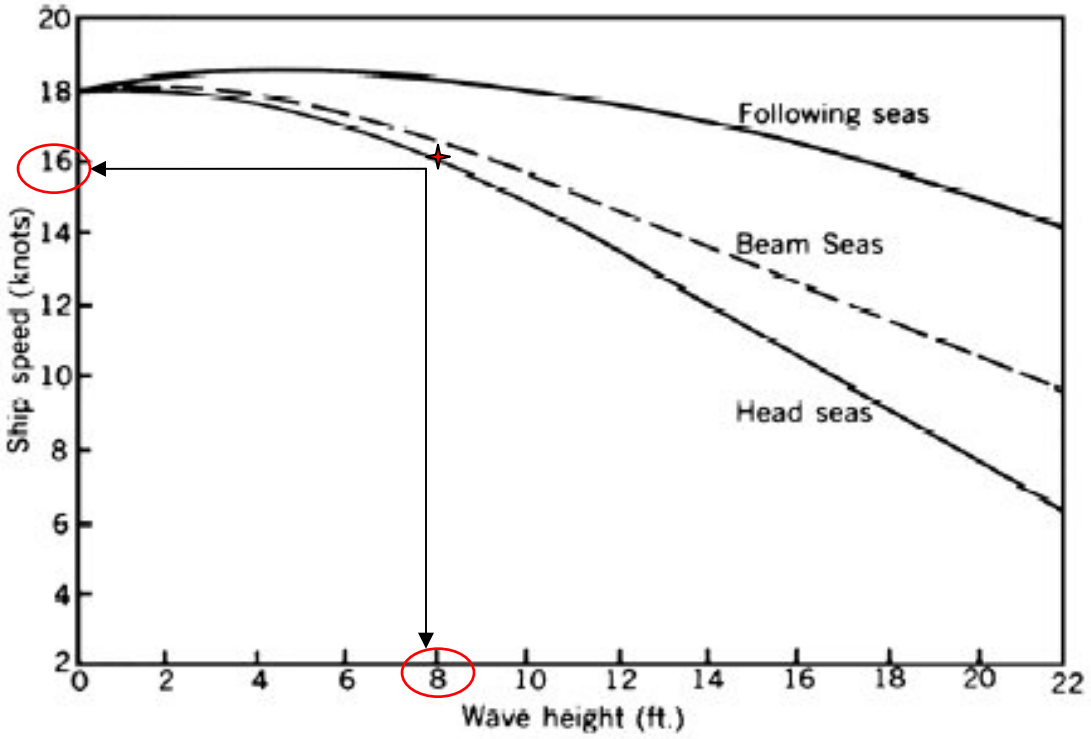
THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX F. FUEL CURVE EXAMPLE



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX G. SPEED REDUCTION CURVES



THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX H. EXAMPLE OF MODEL INPUT

Enter date and zulu time of model run (YYYYMMDDRR format)
2005010112

Enter ship class, i.e. 'FFG7', 'CG47', etc.
CG47

May ship weather limits be slightly exceeded when considering shortest paths? Enter 'Yes' or 'No'.
No

Enter ship's wind limits(knots) in bow, beam, and stern order.
35
35
35

Enter ship's seas limits(feet) in bow, beam, and stern order.
12
12
12

Enter ship's movrep speed in knots to the nearest tenth.
17.0

Enter maximum acceptable speed.
21.0

Enter ship speed to solve shortest path.
17.0

Movement report track input:

Enter the number of waypoints in movement report, including 'start' and 'end' points.
2

Enter Latitude for waypoint 1 in decimal degrees.
39.0
Enter Longitude for waypoint 1 in decimal degrees.
160.0
Enter Latitude for waypoint 2 in decimal degrees.
41.0
Enter Longitude for waypoint 2 in decimal degrees.
163.0

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX I. MODEL ALGORITHM

Index Use

$i \in N$	node in weather network (alias j)
$(i, j) \in A$	arc in weather network $(i, j) \notin bad\_node$
s	start node
t	destination node
f	forecast component (e.g., sea height, direction, …)
a	relative ship aspect (e.g., sea height, wind speed, …)

Given Data

loc_i	location of node i (longitude,latitude,and $forecast\_time_i$)
bad_node_i	node i inaccessible if 1, accessible if 0.
$forecast\_time_i$	time of weather forecast at node i
$wx_{i,f}$	node i forecast component f
$speed$	speed of ship
$ship\_limits_a$	maximum ship-limit for aspect a
$\underline{time}_t, \overline{time}_t$	earliest, latest time of arrival at node t

Working Arrays and Sets

$distance_i$	shortest distance from node s to node i
$time_i$	time from node s to node i
$pred_i$	predecessor of node i in search tree

Functions

transit_distance(i, j)	returns **non-negative** transit distance from node i to node j
$transit\_time(i, j)$	returns transit time from node i to node j
fuel_consumption(s,t)	returns gallons of fuel used from node s to node t
$speed\_penalty(wx_{i,f}, a)$	converts forecast components into ship stress attributes

algorithm *dijkstra*;
begin
$distance_i = \infty \ \forall i \in N$
$time_i = \infty \ \forall i \in N$
$d_s = 0$
$pred(s) = 0$
$HEAP = \{s\}$
while $HEAP \notin \varnothing$ **do**
begin
$i = \underset{j \in HEAP}{\text{argmin}}\{distance_j\}$ (i.e., select i from top of *HEAP*,
move last element in heap to top,
sift down.)
for all $(i, j) \in A$ **do**
begin
if $time_i + transit\_time(i, j) \sim forecast\_time_j$
$\wedge\ time_j > time_i + (transit\_time(i, j) + speed\_penalty(wx_{i,f}, a)$
$\wedge bad\_node_j = 0\ \wedge ship\_limits_a < wx_{i,f}$
then
begin
$time_j = time_i + (transit\_time(i, j) + speed\_penalty(wx_{i,f}, a))$
$distance_j = distance_i + transit\_distance(i, j)$
$pred_j = i$
if $j \notin HEAP$ **then**
Add *j* to end of *HEAP*
Sift *j* up in *HEAP*
end;
end;
end;
end;
end;

Discussion

In addition to ship *speed*, the function *transit_time* considers weather state *wx* and any *ship_limits* that might have influence.

The symbol ”~” indicates ”closest.”

algorithm *arrive on time*;
begin
speed = nominal ship speed
run *dijkstra*
if $time_t < \underline{time}_t$ **then**
$speed = distance_t / \underline{time}_t$
else if $time_t > \overline{time}_t$ **then**
$speed = distance_t / \overline{time}_t$
if *speed* > *maximum allowable speed* **then**
consider storm evasion or delay in port
else
current speed meets arrival criteria
end;
end;

Discussion

This algorithm is an option to the user.

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX J. EXAMPLE OF MODEL OUTPUT

WAYPOINT CALCULATIONS
The distance from point 1 to point 2 is: 182.65 nautical miles.
The bearing from point 1 to point 2 is: 048 degrees True.
The time to transit from point 1 to point 2 at 17.0 kts is: 0010.74 hrs.

MOVREP CALCULATIONS
Total movement report distance is: 0182.65 nm.
Total movement report transit time at 17.0 kts is: 0010.74 hrs.

NETWORK SHORTEST PATH TRACK:
Node: 11899: d[i]:0.0 Latitude = 39.0N Longitude = 160.0E
Node: 12051: d[i]:38.83 Latitude = 39.5N Longitude = 160.5E
Node: 12203: d[i]:77.66 Latitude = 40.0N Longitude = 161.0E
Node: 12053: d[i]:115.5 Latitude = 39.5N Longitude = 161.5E
Node: 12205: d[i]:154.33 Latitude = 40.0N Longitude = 162.0E
Node: 12357: d[i]:192.17 Latitude = 40.5N Longitude = 162.5E
Node: 12509: d[i]:230.01 Latitude = 41.0N Longitude = 163.0E

Elapsed time: 31 ms

SHORTEST PATH CALCULATIONS
Length of shortest path is approximately: 230.01 nm
Distance added to original movement report: 0047.36 nm
Time to complete shortest path at 17.0 kts is 13.53 hrs
Time added to original movement report: 02.79 hrs

FUEL CALCULATIONS
Additional fuel consumption for divert is approximately 3699.30 gallons of F76
Approximate monetary cost is: $4920.07

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX K. MODEL JAVA CODE

```
import java.util.*;
import java.io.*;
import java.text.*;

public class OceanShortestPath {

   // CLASS CONSTANTS

   // This class constant is the cost of 1 gallon of F76 Diesel Fuel Marine for FY05.  Verified by the Manchester Fuel Depot,
   // Puget Sound FISC, 360-476-2135 x310.
   public static final double fuelCost = 1.33;


   // CLASS METHODS

   //Method accesses specific ship class fuel consumption data.  Data is at trail shaft/single engine.  For speeds higher than
   //trail shaft limits, split plant/double engine is used.
   public static int getFuelRate(String shipClass, int speed) {
      int fuelRate;

      int[] ffg = {0,0,0,0,0,373,378,387,391,398,416,438,465,497,531,571,616,670,735,802,876,957,1041,1135,1249
                  ,1395,1568,2065,2314,2790,3100};
      int[] dd = {0,0,0,553,602,651,700,750,822,893,962,1031,1210,1306,1405,1490,1649,1808,1929,2050,2320,2590
                  ,2860,3130,3370,3610,3850,4090,4330,4570,4815};
      int[] ddg = {0,0,0,0,0,583,597,616,640,672,708,754,806,865,934,1013,1103,1204,1317,1444,1731,1893,2062
                   ,2234,2400,3068,3418,3817,4295,4854,5621};
      int[] cg = {0,0,0,0,0,670,690,716,747,785,828,879,935,999,1070,1148,1234,1328,1428,1537,1652,1776,1906,2369
                  ,2547,2739,2948,3178,4010,4307,4649};
      int[] aoe = {0,0,0,0,0,0,0,0,0,0,0,1150,1210,1270,1400,1490,1530,1640,1850,2090,2270,2500,2770,3000,3320,4100
                   ,4630,5200,5600,6190,6520};
      int[] ao = {0,0,0,0,0,0,0,0,293,326,359,391,423,455,500,590,667,745,857,1000,1133,1302,1622,0,0,0,0,0,0,0,0};

      int[] afs = {0,256,257,258,262,268,277,289,306,327,354,388,430,480,540,612,697,797,916,1056,1222,1412,1648
                   ,1922,0,0,0,0,0,0,0};
      int[] lhd = {0,1339,1340,1343,1349,1359,1373,1394,1422,1458,1505,1562,1634,1721,1827,1955,2109,2294,2517
                   ,2786,3112,3507,3989,4581,5311,6221,7362,0,0,0,0};
      int[] lcc = {0,792,792,794,797,802,809,819,833,852,875,905,942,988,1046,1116,1203,1310,1444,1610,1819,2083
                   ,2421,2857,3425,4177,5184,0,0,0,0,0};
      int[] lsd = {0,239,240,241,245,250,259,271,287,307,332,364,401,445,497,556,624,702,789,887,996,1117,1250,1397
                   ,1558,1734,1925,0,0,0,0};

      if (shipClass.equals("FFG7")) {
         fuelRate = ffg[speed];
         return fuelRate;
      }
      else if (shipClass.equals("DD963")) {
         fuelRate = dd[speed];
         return fuelRate;
      }
      else if (shipClass.equals("DDG51")) {
         fuelRate = ddg[speed];
         return fuelRate;
      }
      else if (shipClass.equals("CG47")) {
         fuelRate = cg[speed];
         return fuelRate;
      }
      else if (shipClass.equals("AOE6")) {
         fuelRate = aoe[speed];
         return fuelRate;
      }
      else if (shipClass.equals("AO187")) {
         fuelRate = ao[speed];
         return fuelRate;
      }
```

```
    else if (shipClass.equals("AFS1")) {
        fuelRate = afs[speed];
        return fuelRate;
    }
    else if (shipClass.equals("LHD1")) {
        fuelRate = lhd[speed];
        return fuelRate;
    }
    else if (shipClass.equals("LCC19")) {
        fuelRate = lcc[speed];
        return fuelRate;
    }
    else if (shipClass.equals("LSD41")) {
        fuelRate = lsd[speed];
        return fuelRate;
    }
    else {
        return 0;
    }
}

//Method comoutes fuel costs.
public static double getFuelCost(int rate, double divTime) {
    double cost = fuelCost*rate*divTime;
    return cost;
}

//Method computes arc distance based on node number
public static int getArcCost(int i, int j) {
    //All lengths in 100ths of a nm
    int [] diagLength = {4247,4239,4215,4175,4121,4053,3973,3883,3784,3679,3570};
    int [] latLength  = {3006,2995,2961,2904,2826,2726,2605,2465,2306,2129,1936};
    int longLength = 3006;

    int latBlock;

    latBlock=(i-1)/1510;
    if(latBlock>=11){
        System.out.println("latBlock "+latBlock+"  i "+i);
        return 0;
    }

    if(j == i-152 || j == i+152 || j == i-150 || j == i+150) {
        return diagLength[latBlock];
    }
    else if(j == i-1 || j == i+1) {
        return latLength[latBlock];
    }
    else if(j == i-151 || j == i+151) {
        return longLength;
    }
    else {
        System.out.println("Nodes not connected in graph: i="+i+" j="+j);
    }
    return 0;
}

//Method will calculate ship's course based on head and tail nodes for network graph.
public static int getNetworkCourse(int i, int j) {
    int course;
    if(i + 151 == j) {
        course = 0;
        return course;
    }
    else if (i + 152 == j) {
        course = 45;
        return course;
    }
    else if (i + 1 == j) {
        course = 90;
```

```
        return course;
    }
    else if (i - 150 == j) {
        course = 135;
        return course;
    }
    else if (i - 151 == j) {
        course = 180;
        return course;
    }
    else if (i - 152 == j) {
        course = 225;
        return course;
    }
    else if (i - 1 == j) {
        course = 270;
        return course;
    }
    else if (i + 150 == j) {
        course = 315;
        return course;
    }
    else {
        return 0;
    }
}


//Method converts Latitude and Longitude into corresponding node number
public static double getNodeNumber(double lat, double lon) {
    double nodeNumber;
    //This will round lat and lon to nearest half degree node.
    if (lat - Math.floor(lat) == 0) {
        lat = lat;
    }
    else if (lat - Math.floor(lat) > 0 && lat - Math.floor(lat) <= .24) {
        lat = Math.floor(lat);
    }
    else if (lat - Math.floor(lat) > .24 && lat - Math.floor(lat) <= .49) {
        lat = Math.floor(lat) + .5;
    }
    else if (lat - Math.floor(lat) == .5) {
        lat = lat;
    }
    else if (lat - Math.floor(lat) > .5 && lat - Math.floor(lat) <=.74) {
        lat = Math.floor(lat) + .5;
    }
    else if (lat - Math.floor(lat) > .74 && lat - Math.floor(lat) <= .99) {
        lat = Math.floor(lat) + 1.0;
    }
    if (lon - Math.floor(lon) == 0) {
        lon = lon;
    }
    else if (lon - Math.floor(lon) > 0 && lon - Math.floor(lon) <= .24) {
        lon = Math.floor(lon);
    }
    else if (lon - Math.floor(lon) > .24 && lon - Math.floor(lon) <= .49) {
        lon = Math.floor(lon) + .5;
    }
    else if (lon - Math.floor(lon) == .5) {
        lon = lon;
    }
    else if (lon - Math.floor(lon) > .5 && lon - Math.floor(lon) <=.74) {
        lon = Math.floor(lon) + .5;
    }
    else if (lon - Math.floor(lon) > .74 && lon - Math.floor(lon) <= .99) {
        lon = Math.floor(lon) + 1.0;
    }

    nodeNumber = ((2 * lat) * 151) + ((2 * (lon - 100)) + 1);
```

```
    return nodeNumber;
}

//Method converts node number to corresponding Latitude
public static double getLatitude(int nodeNumber) {
    double latitude;
    latitude = (Math.floor((nodeNumber - 1)/ 151))/2;
    return latitude;
}

//Method converts node number to corresponding Longitude
public static double getLongitude(int nodeNumber, double latitude) {
    double longitude;
    longitude = (.5 * (nodeNumber - (2 * latitude * 151) -1)) + 100;
    return longitude;
}

//Method calculates time to complete distance given ship's speed
public static double getTimeDouble(double distance, double shipSpeed) {
    double time;
    time = ((distance)/shipSpeed);
    return time;
}

//Method retrieves speed penalty for transiting an arc that exceeds ship limits
public static int getSpeedPenalty(int aspect, int seaHeight) {
    int [][] penalty = {{0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0,0},
    {0,0,0,0,0,-1,-1,-1,-2,-3,-3,-4,-4,-5,-6,-7,-7,-8,-9,-9,-11,-11,-12},
    {0,0,0,0,0,0,0,0,-1,-1,-1,-2,-2,-3,-3,-4,-4,-5,-5,-6,-6,-7,-7,-8},
    {0,0,0,0,1,1,1,1,1,1,1,0,0,0,0,-1,-1,-1,-2,-2,-2,-3,-3}};
    if(aspect < 0 || seaHeight < 0) {
        return -100;
    }
    else {
        return penalty[aspect][seaHeight];
    }
}

public static int getFcstTime(double time) {
    int fcstTime;
    if (time > 75.0) {
        return fcstTime = 12;
    }
    else if (time/6 < Math.floor(time/6) + .5) {
        return fcstTime = (int)Math.floor(time/6);
    }
    else {
        return fcstTime = (int)Math.floor(time/6) + 1;
    }
}


//Method calculates relative bearing of wind or seas given ship's true course and true bearing of wind and seas.
public static int calcRB(int wxBearing, int course) {
    int relBrng;
    relBrng = wxBearing - course;
    if (relBrng < 0) {
        relBrng += 360;
    }
    return relBrng;
}

//Method takes numerical relative bearing and correlates it to the ship's hull position.  1 = Bow, 2 = Beam, 3 = Stern.
public static int getRBPos(int relBrng) {
    if (relBrng>= 0 && relBrng <= 45) {
        return 1;
    }
    else if (relBrng > 45 && relBrng < 135) {
        return 2;
    }
```

```
    else if (relBrng >=135 && relBrng <=225) {
        return 3;
    }
    else if (relBrng > 225 && relBrng < 315) {
        return 2;
    }
    else if (relBrng <= 315 && relBrng <360) {
        return 1;
    }
    else {
        return 0;
    }
}

//Calculates course from start point to end point.
public static double courseCalc(double lat1, double lon1, double lat2, double lon2, double distance) {
    double bearing;
    lat1 = (lat1/180.0)*Math.PI;
    lat2 = (lat2/180.0)*Math.PI;
    lon1 = (lon1/180.0)*Math.PI;
    lon2 = (lon2/180.0)*Math.PI;
    distance = ((distance*Math.PI)/(.539612*1.852*60*180.0));

    double t1 = Math.sin(lat2) - Math.sin(lat1) * Math.cos(distance);
    double t2 = Math.cos(lat1) * Math.sin(distance);
    double t3 = t1/t2;
    double t4 = -t3*t3+1;

    if (Math.sin(lon2 - lon1) < 0.0) {
        bearing = Math.atan(-t3/Math.sqrt(-t3*t3+1)) + 2*Math.atan(1);
    }
    else {
        bearing = 2 * 3.14 - (Math.atan(-t3/Math.sqrt(t4)) + 2*Math.atan(1));

    }
    return 360.0 - (bearing/Math.PI)*180.0;

}

//Calculates distance between two geographical points.
public static double distanceCalc(double lat1, double lon1, double lat2, double lon2) {
    double distance;
    lat1 = (lat1/180.0)*Math.PI;
    lat2 = (lat2/180.0)*Math.PI;
    lon1 = (lon1/180.0)*Math.PI;
    lon2 = (lon2/180.0)*Math.PI;

    distance = Math.acos(Math.sin(lat1)*Math.sin(lat2) + Math.cos(lat1)*Math.cos(lat2)*Math.cos(lon2-lon1));
    return .539612*1.852*60*(distance/Math.PI)*180.0;
}

 public static double distCalc(int i, int j) {
    double distance;
    double lat1 = (getLatitude(i)/180.0)*Math.PI;
    double lat2 = (getLatitude(j)/180.0)*Math.PI;
    double lon1 = (getLongitude(i, lat1)/180.0)*Math.PI;
    double lon2 = (getLongitude(j, lat2)/180.0)*Math.PI;

    distance = Math.acos(Math.sin(lat1)*Math.sin(lat2) + Math.cos(lat1)*Math.cos(lat2)*Math.cos(lon2-lon1));
    return .539612*1.852*60*(distance/Math.PI)*180.0;
}

public static boolean exceedOrNot(int windSpeed, int seaHeight, int shipWindLim, int shipSeaLim) {
    if (windSpeed > shipWindLim || seaHeight > shipSeaLim) {
        return true;
    }
    else {
        return false;
    }
}
```

```
 /**
 * Main method
 * @param args Command line arguments
 */
public static void main(String[] args) {
   // Text Input Tools
   String         inputString;
   BufferedReader  inputUnit;
   StringTokenizer tk;
   String         weatherFilename;
   String         dateStamp;

   //Forward Star data structure for Ocean Network
   int rows = 101;
   int cols = 151;
   int C=4247;
   int i,j,m,tc;
   int n = rows*cols;
   int mmax = 4*3 + 2*(rows-2 + cols-2)*5 + (rows-2)*(cols-2)*8;
   int node;
   int[] point;
   int[] tail;
   int[] head;
   int[] cost;
   int[] badNode;

   point = new int[n+2];
   tail = new int[mmax+1];
   head = new int[mmax+1];
   cost = new int[mmax+1];
   badNode =  new int[n+1];

   //Weather parameter arrays for tau 00 to 72 hour in 6 hour time steps.
   int[][][]windParam;
   int[][][]seasParam;

   //[timestamp][table wind, (dir=0, spd=1) or seas, (hgt=0, dir=1)][node]*//
   windParam = new int [13][2][n+2];
   seasParam = new int [13][2][n+2];

   //Read weather parameter text data into weather parameter arrays.
   System.out.println("Enter date and zulu time of model run (YYYYMMDDRR format)");
   String date = SavitchIn.readLine();

   for(int tm=0;tm<13;tm++){
      weatherFilename="C:/Ops Research/THESIS/"+date+".F";
      if(tm<2)
         weatherFilename=weatherFilename+"00"+Integer.toString(6*tm)+".txt";
      else
         weatherFilename=weatherFilename+"0"+Integer.toString(6*tm)+".txt";
      try {
         inputUnit = new BufferedReader(new FileReader(weatherFilename));

         inputString = inputUnit.readLine();  //Skip first two header lines
         inputString = inputUnit.readLine();
         node=0;
         while((inputString = inputUnit.readLine())!=null){
            node++;
            tk = new StringTokenizer(inputString);
            tk.nextToken();  //Skip first two tokens
            tk.nextToken();
            windParam[tm][0][node] = Integer.parseInt(tk.nextToken());
            windParam[tm][1][node] = Integer.parseInt(tk.nextToken());
            seasParam[tm][0][node] = Integer.parseInt(tk.nextToken());
            seasParam[tm][1][node] = Integer.parseInt(tk.nextToken());

         }
         inputUnit.close();
```

```
    } catch(FileNotFoundException e) {
        System.out.println(e);
        return;
    } catch(IOException e){
        System.out.println(e);
        return;
    } catch(NoSuchElementException e) {
        System.out.println("No data found");
        return;
    }

}


//Read geographically inaccessible nodes into graph
int k=0;
try {
    inputUnit = new BufferedReader(new FileReader("C:/Ops Research/THESIS/Bad_Nodes.txt"));
    while((inputString = inputUnit.readLine())!=null){
        k++;
        node = Integer.parseInt(inputString);
        if(node<1 || node >n){
            System.out.println("Illegal node number in bad nodes file at line "+k+": node="+node);
            return;
        }
        badNode[node]=1;
    }
} catch(FileNotFoundException e) {
    System.out.println(e);
    return;
} catch(IOException e){
    System.out.println(e);
    return;
} catch(NoSuchElementException e) {
    System.out.println("No data found: k="+k);
    return;
}

point[1] = 1;
m=1;
i=1;
try{
    for(i = 1; i <= n; i++) {
        //Creates Ocean Graph
        //Corner nodes
        if(i==1) {
            point[i+1] = point[i] + 3;
            tail[m] = i;
            head[m] = i + 1;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i + 151;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i + 152;
            cost[m] = getArcCost(i,head[m]);
            m++;
        } else if(i==151) {
            point[i+1] = point[i] + 3;
            tail[m] = i;
            head[m] = i - 1;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i + 151;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
```

```
    head[m] = i + 150;
    cost[m] = getArcCost(i,head[m]);
    m++;
  } else if(i==15100) {
    point[i+1] = point[i] + 3;
    tail[m] = i;
    head[m] = i + 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 150;
    cost[m] = getArcCost(i,head[m]);
    m++;
  } else if(i==15251) {
    point[i+1] = point [i] + 3;
    tail[m] = i;
    head[m] = i - 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 152;
    cost[m] = getArcCost(i,head[m]);
    m++;
  }
  //Bottom Edge nodes
  else if(i>1 & i<151) {
    point[i+1] = point [i] + 5;
    tail[m] = i;
    head[m] = i + 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 150;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 152;
    cost[m] = getArcCost(i,head[m]);
    m++;
  }
  //Top Edge nodes
  else if(i>15100 & i<15251) {
    point[i+1] = point [i] + 5;
    tail[m] = i;
    head[m] = i + 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i -150;
    cost[m] = getArcCost(i,head[m]);
```

```
    m++;
    tail[m] = i;
    head[m] = i - 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 152;
    cost[m] = getArcCost(i,head[m]);
    m++;
}
//Right Edge nodes
else if(i%151 == 0) {
    point[i+1] = point [i] + 5;
    tail[m] = i;
    head[m] = i - 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 150;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 152;
    cost[m] = getArcCost(i,head[m]);
    m++;
}
//Left Edge nodes
else if((i-1)%151 == 0) {
    point[i+1] = point [i] + 5;
    tail[m] = i;
    head[m] = i + 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 151;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 152;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i - 150;
    cost[m] = getArcCost(i,head[m]);
    m++;
}
//All other nodes
else {
    point[i+1] = point [i] + 8;
    tail[m] = i;
    head[m] = i - 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
    head[m] = i + 1;
    cost[m] = getArcCost(i,head[m]);
    m++;
    tail[m] = i;
```

```
            head[m] = i - 151;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i + 151;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i - 152;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i + 152;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i - 150;
            cost[m] = getArcCost(i,head[m]);
            m++;
            tail[m] = i;
            head[m] = i + 150;
            cost[m] = getArcCost(i,head[m]);
            m++;
         }
      }
   } catch(ArrayIndexOutOfBoundsException e){
      System.out.println(e+" i "+i+" m "+m);
   }

   /*
   //Network Graph Output.
   for(i=1; i <= 10; i++){
      System.out.println(i + "    "  + point[i] + "  Bad Node? "+badNode[i]);
      for(int p = point[i]; p < point[i+1]; p++) {
         System.out.println("     " + tail[p]+"        "+head[p]+"         "+cost[p]);
      }
   }
    */



   //Dijkstra 2 HEAP data structure elements
   int t, nC;
   int s;
   double transitTime;
   int [] pred;
   int [] d;
   double [] time;

   pred   = new int [n+1];
   d      = new int [n+1];
   time   = new double [n+1];

   BinaryHeap H = new BinaryHeap(n+1);



   //Format starting Latitude and Logitude input and display corresponding nodenumber.
   DecimalFormat pointTwoDigits = new DecimalFormat("0000.00");
   DecimalFormat twoDigits = new DecimalFormat("00.00");
   DecimalFormat degrees = new DecimalFormat("000");

   //Input ship class.
   System.out.println("Enter ship class, i.e. 'FFG7', 'CG47', etc.");
   String shipClass = SavitchIn.readLine();

   //Input ship weather limits, {0 = wind (1 = bow, 2 = beam, 3 = stern), 1 = seas(1 = bow, 2 = beam, 3 = stern)}
   String limitDec;
   System.out.println("May ship weather limits be slightly exceeded when considering shortest paths? Enter 'Yes' or 'No'.");
   limitDec = SavitchIn.readLine();
```

```
        int shipParam[][];
        shipParam = new int [2][4];
        System.out.println("Enter ship's wind limits(knots) in bow, beam, stern order.");
        for (int b = 1; b <=3; b++) {
           shipParam[0][b] = SavitchIn.readLineInt();
        }
        System.out.println("Enter ship's seas limits(feet) in bow, beam, stern order.");
        for (int b = 1; b <=3; b++) {
           shipParam[1][b] = SavitchIn.readLineInt();
        }

        //Input Speed of Advance (SOA).
        System.out.println("Enter ship's movrep speed in knots to the nearest tenth.");
        double movSpeed = SavitchIn.readLineDouble();
        System.out.println("Enter maximum acceptable speed.");
        double maxSpeed = SavitchIn.readLineDouble();
        System.out.println("Enter ship speed to solve shortest path.");
        double speed = SavitchIn.readLineDouble();

        //Input Movrep track
        System.out.println("MOVREP track input:");
        System.out.println("");
        System.out.println("Enter the number of waypoints in MOVREP track, including 'start' and 'end' points.");
        int numWP = SavitchIn.readLineInt();


        //Calculate Movrep data
        double[] lats;
        double[] longs;
        double[] movDist;
        lats = new double[numWP+1];
        longs = new double[numWP+1];
        movDist = new double[numWP+1];

        for (int w = 1; w <= numWP; w++) {
           System.out.println("Enter Latitude for waypoint "+w+" in decimal degrees.");
           lats[w] = SavitchIn.readLineDouble();
           System.out.println("Enter Longitude for waypoint "+w+" in decimal degrees.");
           longs[w] = SavitchIn.readLineDouble();
        }
        for (int x = 1; x < numWP; x++) {
           double D = distanceCalc(lats[x], longs[x], lats[x+1], longs[x+1]);
           movDist[x] = D;
           System.out.println("");
           System.out.println(" WAYPOINT CALCULATIONS ");
           System.out.println(" The distance from point "+x+" to point "+(x+1)+" is: "+twoDigits.format(D)+" nautical miles.");
           System.out.println(" The bearing from point "+x+" to point "+(x+1)+" is: "+degrees.format(courseCalc(lats[x], longs[x],
lats[x+1], longs[x+1], D))+" degrees True.");
           System.out.println(" The time to transit from point "+x+" to point "+(x+1)+" at "+speed+" kts is:
"+pointTwoDigits.format(getTimeDouble(D, movSpeed))+" hrs.");
           System.out.println("");
        }
        double totalMovDist = 0.0;
        for(int index = 0; index < movDist.length; index++) {
           totalMovDist += movDist[index];
        }
        System.out.println(" MOVREP CALCULATIONS ");
        System.out.println(" Total movrep distance = "+pointTwoDigits.format(totalMovDist)+" nm.");
        double totalMovTime = totalMovDist/movSpeed;
        System.out.println(" Total movrep transit time at "+movSpeed+" kts is: "+pointTwoDigits.format(totalMovTime)+" hrs.");

        //Initialize s and t nodes.
        double startLat = lats[1];
        pointTwoDigits.format(startLat);
        double startLong = longs[1];
        pointTwoDigits.format(startLong);

        s = (int)getNodeNumber(startLat, startLong);

        double endLat = lats[numWP];
```

```
pointTwoDigits.format(endLat);
double endLong = longs[numWP];
pointTwoDigits.format(endLong);

t = (int)getNodeNumber(endLat, endLong);


long startTime, elapsedTime;
startTime=System.currentTimeMillis();

//Dijkstra 2 HEAP algorithm
if((n*C)/n==C)
   nC=n*C;
else
   nC=1000000000;

for(i=1; i<=n; i++) {
   d[i]=nC;
   time[i] = nC;
}
d[s]=0;
time[s]=0;
pred[s]=0;
H.insert(s,d[s]);

if (limitDec.equals("No") || limitDec.equals("no")) {

   while (H.inheap() > 0) {
      i = H.findMin();
      H.deleteMin();    //Select i from top of HEAP, move last element in HEAP to top, then sift down.
      if (i == t) {
         break;
      }
      for (k = point[i]; k < point[i+1]; k++) {
         j = head[k];
         int course = getNetworkCourse(i, j);   //Initialize and calculate all variables necessary for time label update.
         double tripTime = time[i] + (distCalc(i, j)/speed);
         int fcstTime = getFcstTime(tripTime);
         int windDir = windParam[fcstTime][0][j];
         int seaDir = seasParam[fcstTime][1][j];
         int windAspect = getRBPos(calcRB(windDir, course));
         int seaAspect = getRBPos(calcRB(seaDir,course));
         int windSpeed = windParam[fcstTime][1][j];
         int seaHeight = seasParam[fcstTime][0][j];
         int shipWindLim = shipParam[0][windAspect];
         int shipSeaLim = shipParam[1][seaAspect];
         if(badNode[j]==0 && time[j] > time[i] + (cost[k]/(speed + getSpeedPenalty(seaAspect, seaHeight)))/100.0 &&
exceedOrNot(windSpeed, seaHeight, shipWindLim, shipSeaLim)== false) {
            time[j] = time[i] + (cost[k]/(speed + getSpeedPenalty(seaAspect, seaHeight)))/100.0;
            d[j] = d[i] + cost[k];
            pred[j] = i;
            if (H.onheap(j) == 0) {
               H.insert(j, d[j]);
            }
            else {
               H.decreaseKey(j, d[j]);  //If j is not an element of HEAP, add j to end of HEAP, sift j up.
            }
         }
      }
   }
}
else {
   while (H.inheap() > 0) {
      i = H.findMin();
      H.deleteMin();    //Select i from top of HEAP, move last element in HEAP to top, then sift down.
      if (i == t) {
         break;
      }
      for (k = point[i]; k < point[i+1]; k++) {
         j = head[k];
```

```
            int course = getNetworkCourse(i, j);
            double tripTime = time[i] + (distCalc(i, j)/speed);
            int fcstTime = getFcstTime(tripTime);
            int windDir = windParam[fcstTime][0][j];
            int seaDir = seasParam[fcstTime][1][j];
            int windAspect = getRBPos(calcRB(windDir, course));
            int seaAspect = getRBPos(calcRB(seaDir,course));
            int windSpeed = windParam[fcstTime][1][j];
            int seaHeight = seasParam[fcstTime][0][j];
            int shipWindLim = shipParam[0][windAspect];
            int shipSeaLim = shipParam[1][seaAspect];
            if(badNode[j]==0 && time[j] > time[i] + (cost[k]/(speed + getSpeedPenalty(seaAspect, seaHeight)))/100.0) {
               time[j] = time[i] + (cost[k]/(speed + getSpeedPenalty(seaAspect, seaHeight)))/100.0;
               d[j] = d[i] + cost[k];
               pred[j] = i;
               if (H.onheap(j) == 0) {
                  H.insert(j, d[j]);
               }
               else {
                  H.decreaseKey(j, d[j]);  //If j is not an element of HEAP, add j to end of HEAP, sift j up.
               }
            }
         }
      }
   }

elapsedTime=System.currentTimeMillis()-startTime;

//Display output
i=t;
int [] stack;
stack = new int [n];
int top = 0;
while(i>0){
   stack[top++]=i;
   i=pred[i];
}
System.out.println("");
System.out.println(" NETWORK SHORTEST PATH TRACK:");
while(top>0){
   i=stack[--top];
   System.out.print(" Node: "+i+":  d[i]:"+d[i]/100.0);
   System.out.println("    Latitude = "+getLatitude(i)+"N  Longitude = "+getLongitude(i, getLatitude(i))+"E");
}

System.out.println("");
System.out.println(" Elapsed time: " + elapsedTime + " ms");
System.out.println("");

//Output divert information.
double spDist = d[t]/100.0;
double spTime = getTimeDouble(spDist, speed);
double divDist = spDist - totalMovDist;
double divertTime = getTimeDouble(divDist, speed);
System.out.println("");
System.out.println(" SHORTEST PATH CALCULATIONS ");
System.out.println(" Length of shortest path is approximately: "+spDist+" nm ");
System.out.println(" Distance added to original movrep: "+pointTwoDigits.format(divDist)+" nm ");
System.out.println(" Time to complete shortest path at "+speed+" kts is "+twoDigits.format(spTime)+" hrs");
System.out.println(" Time added to original movrep: "+twoDigits.format(divertTime)+" hrs");
System.out.println("");

//Calculate Metrics.
int spd = (int) speed;
int rate = getFuelRate(shipClass, spd);
double fuelUsed = (divertTime*rate);
double totalCost = getFuelCost(rate, divertTime);

//Output Fuel Metrics.
System.out.println(" FUEL CALCULATIONS ");
```

```
        System.out.println(" Additional fuel consumption for divert is approximately "+pointTwoDigits.format(fuelUsed)+" gallons of
F76");
        System.out.println(" Approximate monetary cost is: $"+pointTwoDigits.format(totalCost));
        System.out.println("");

        //Calculate arrival time metrics.
        /*
        double newSpeed;
        System.out.println("");
        System.out.println(" ARRIVAL TIME CALCULATIONS ");
        System.out.println(" Enter number of hours ship can arrive early at its destination.");
        int earlyDelta = SavitchIn.readLineInt();
        System.out.println(" Enter number of hours ship can arrive late at its destination.");
        int lateDelta = SavitchIn.readLineInt();

        if(spTime > totalMovTime + lateDelta) {
           System.out.println(" At current speed, ship arrives at destination too late.");
           newSpeed = spDist/(totalMovTime + lateDelta);
           if (newSpeed >= maxSpeed){
              System.out.println(" Recommended speed to arrive at latest allowable time is in excess of 25 kts.  " +
              "It is possible weather will not permit transit with current time constraints." +
              "Recommend reduction in SOA, evasion, or delay in port");
           }
           else {
              System.out.println(" Recommend repeating shortest path algorithm using "+twoDigits.format(newSpeed)+" kts.");
           }
        }
        else if (spTime < totalMovTime - earlyDelta) {
           System.out.println(" At current speed, ship arrives at destination too early");
           newSpeed = spDist/(totalMovTime - earlyDelta);
           System.out.println(" Recommend repeating shortest path algorithm using "+twoDigits.format(newSpeed)+" kts.");
        }
        else {
           System.out.println(" Current speed meets time criteria.");
        }
         */
    }
}
```

LIST OF REFERENCES

Ahuja, R., et al. Network Flows, Theory, Algorithms, and Applications. New Jersey: Prentice Hall, 1993.

American Society of Naval Engineers [2004] "Archives of Past Meeting," http://www.navalengineers.org/Sections/GoldenGate/mtgs0003.html, accessed 17 December.

Brown, J. L. (1993). "A Cost Benefit Analysis of Two Products of the Fleet Numerical Oceanography Center," Thesis, Naval Postgraduate School, Monterey, CA.

Department of the Army, UNC/CFC/USFK Public Affairs Office [2004] "U.S. Logistics Vessel Runs Aground," http://www.korea.army.mil/PAO/news/010506.htm, accessed 17 December.

Department of the Navy – Naval Meteorology and Oceanography Command Instruction 3140.1L [2000] United States Navy Meteorological and Oceanographic Support System Manual, Naval Meteorology and Oceanography Command, Stennis Space Center, MS.

Department of the Navy – Naval Warfare Publication 1-03.1(IC-1) (formerly Naval Warfare Publication 10-1-10) [1997] Operational Reports, Naval Doctrine Command, Norfolk, VA.

Department of the Navy, Fleet Numerical Meteorology and Oceanography Center [2005] "Model Characteristics and Tendencies for NOGAPS 4.0, COAMPS 3.0, and WW3 1.18," http://www.fnmoc.navy.mil/PUBLIC, accessed 4 June.

Department of the Navy, Naval Historical Center [2004] "Typhoons and Hurricanes: Pacific Typhoon, 18 December 1944," http://www.history.navy.mil/faqs/faq102-4.htm, accessed 17 December.

Fagerholt, K., et al. (2000). "Shortest Path in the Presence of Obstacles: An Application to Ocean Shipping," Journal of Operational Research Society, 51, pp. 682-688.

Lee, H., et al. (2002). "Optimum Ship Routing and its Implementation on the Web," Lecture Notes in Computer Science, Advanced Internet Services and Applications: First International Workshop, 2402, pp. 125-136.

National Geospatial Intelligence Agency [2004] "Principles of Weather Routing," http://164.214.12.145/NAV_PUBS/APN/Chapt-38.pdf, accessed 14 December.

Naval Sea Systems Command [2005] "NAVSEA Incentivized Energy Conservation Program," http://www.i-encon.comm, accessed 3 February.

Ocean Systems Incorporated [2005] "Recent Technologies for the Maritime Industry," http://www.ocean-systems.com, accessed 14 January.

Savitch, Walter. Java, An Introduction to Computer Science & Programming. New Jersey: Prentice Hall, 2001.

Schrady, D., et al. (1996). “Predicting Ship Fuel Consumption: Update,” Naval Postgraduate School, Monterey, CA.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California

3. Professor Gerald G. Brown
   Naval Postgraduate School
   Monterey, California

4. Professor W. Matthew Carlyle
   Naval Postgraduate School
   Monterey, California

5. Commanding Officer
   Fleet Numerical Meteorology and Oceanography Center
   Monterey, California

6. Commanding Officer
   Naval Pacific Meteorology and Oceanography Center
   Pearl Harbor, Hawaii

7. LT Anel A. Montes, USN
   Monterey, California